Privacy and the Complexity of Simple Queries

Privacy and the Complexity of Simple Queries

A dissertation presented
by

Jonathan Robert Ullman

to
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
May 2013

Dissertation Advisor: Salil P. Vadhan Jonathan Robert Ullman

Privacy and the Complexity of Simple Queries

Abstract

As both the scope and scale of data collection increases, an increasingly large amount of sensitive personal information is being analyzed. In this thesis, we study the feasibility of effectively carrying out such analyses while respecting the privacy concerns of all parties involved. In particular, we consider algorithms that satisfy *differential privacy* [30], a stringent notion of privacy that guarantees no individual's data has a significant influence on the information released about the database. Over the past decade, there has been tremendous progress in understanding when accurate data analysis is compatible with differential privacy, with both elegant algorithms and striking impossibility results. However, if we ask further when accurate and *computationally efficient* data analysis is compatible with differential privacy then our understanding lags far behind. In this thesis, we make several contributions to understanding the complexity of differentially private data analysis:

- We show a sharp upper bound on the number of linear queries that can be accurately answered while satisfying differential privacy by an *efficient* algorithm, assuming the existence of cryptographic *traitor-tracing schemes*.
- We show even stronger computational barriers for algorithms that generate private *synthetic data*—a new database that consists of "fake" records but preserves certain statistical properties of the original database. Under cryptographic assumptions, any efficient differentially private algorithm that generates synthetic data cannot preserve even extremely simple properties of the database, even the pairwise correlations between the attributes.
- On the positive side, we design new algorithms for the widely-used class of *marginal queries* that are faster and require less data.

Computational inefficiency is not the only barrier to effective privacy-preserving data analysis. Another potential obstacle is that many of the existing differentially private algorithms do not

guarantee *privacy for the data analyst*, which would lead researchers with sensitive or proprietary queries to seek other means of access to the database. We also contribute to our understanding of privacy for the analyst:

- We design new algorithms for answering large sets of queries that guarantee differential privacy for the database and ensure differential privacy for the analysts, even if all other analysts collude.

Contents

Abstract . . . iii
Table of Contents . . . v
Acknowledgments . . . vii

1 Introduction **1**
1.1 Contributions of this Thesis . . . 4

2 Background **8**
2.1 Databases . . . 8
2.2 Differential Privacy . . . 8
2.3 Sanitizers and Counting Queries . . . 11
2.4 Some Differentially Private Algorithms . . . 14

3 Answering $\mathbf{n^{2+o(1)}}$ Arbitrary Counting Queries is Hard **24**
3.1 Results and Techniques . . . 24
3.2 Traitor-Tracing Schemes . . . 29
3.3 Attacking Efficient Sanitizers . . . 32
3.4 Constructions of Traitor-Tracing Schemes . . . 35

4 The Hardness of Generating Private Synthetic Data **50**
4.1 Our Results and Techniques . . . 52
4.2 Relationship with Hardness of Approximation . . . 55
4.3 Hard-to-Sanitize Distributions from Hard CSPs . . . 62
4.4 Relaxed Synthetic Data . . . 72

5 Faster Algorithms for Privately Releasing Marginal Queries **87**
5.1 Our Results and Techniques . . . 88
5.2 Preliminaries . . . 92
5.3 From Low-Dimensional Approximations to One-Shot Sanitizers . . . 94
5.4 Low-Dimensional Approximations . . . 97

6 Faster Algorithms for Marginal Queries on Small Databases **103**
6.1 Introduction . . . 103

6.2 Preliminaries . . . 108
6.3 From Low-Weight Approximations to Sanitizers . . . 109
6.4 Low-Weight Approximations . . . 113
6.5 Limitations of Low-Weight Approximations . . . 120

7 New Analyst-Private Algorithms **129**
7.1 Introduction . . . 129
7.2 Preliminaries . . . 134
7.3 Solving Two-Player Zero-Sum Games . . . 136
7.4 A One-Query-to-Many-Analyst Private Sanitizer . . . 140
7.5 A One-Analyst-to-Many-Analyst Private Sanitizer . . . 149
7.6 Another One-Query-to-Many-Analyst Private Sanitizer . . . 154
7.7 A Secrecy-of-the-Sample Lemma . . . 164

8 Conclusion **167**

Bibliography **170**

Acknowledgments

I am extremely fortunate to have had Salil as my advisor. Salil is an amazing teacher, a brilliant researcher, and an excellent mentor. But I'll always be most amazed by how he sees things so clearly that the right answer always seems obvious. Salil likes to joke that I never work with him anymore, and prefer to work by myself or with other collaborators. The reality is quite the opposite—I still can't imagine venturing into the world of research without his guidance. When in doubt, I will always ask myself what Salil would do.

In addition to Salil, my time at Harvard has been shaped by many fantastic people. I've had the good fortune to call Maxwell-Dworkin 138 my office, and share it with a fun cast of characters: Kai-Min, Zhenming, Anna, Colin, Justin, Varun, Thomas, Scott, Jiayiang, Jiapeng, Mark, Tom. I thank them, and all the honorary MD138'ers for making sure I remembered to have fun. I would like to thank each of my thesis committee members, Kobbi Nissim, Michael Mitzenmacher, and David Parkes. And I also would like to thank Les Valiant and Radhika Nagpal, who served on my qualifying committee. Carol Harlow has been instrumental in maintaining my sanity by covering for my inability to do paperwork, and I sincerely thank her for that.

Although he has refused to accept the distinction, Boaz Barak is the reason I went to graduate school. He "agreed" to advise my undergraduate senior thesis, introduced me to differential privacy, and treated me like I belonged in research well before I deserved it. Boaz also introduced me to Adam Smith, who hosted me for an eye-opening trip to Penn State during my senior thesis, which I am greatly thankful for. I also want to thank Cynthia Dwork who hosted me for a fantastic internship at Microsoft Research. Cynthia's talent and enthusiasm for research is a constant inspiration. I have been fortunate to work with many great researchers, and I thank them all for the great experiences. I would especially like to thank Moritz Hardt and Aaron Roth. Moritz and Aaron have always given me great inspiration, great advice, and been great friends.

Harvard would have been a dreary experience without my girlfriend Hilary. Her support keeps me going, her talents inspire me, and her velociraptor sounds both terrify and entertain me.

Anything I could say about my family in this space would be underwhelming. So I will simply say thank you for always letting me figure it out on my own.

Chapter 1

Introduction

Wider collection of data is predicted to have a transformative effect in areas as diverse as healthcare, education, government, and online commerce (see [48, 63, 56] for a few examples of such claims), and research in statistics, machine learning, and the social sciences is rapidly generating more powerful data analysis techniques to harness the commercial and social value of this data. However, many of the richest and most important data sets are comprised of sensitive, private information. The result is a dilemma: allowing unrestricted access to and sharing of this data might be unethical, illegal, or reputationally damaging for the data collectors, holders, and analysts, and this harm cuts against the social benefit of analyzing the data. On the other hand, access to the data can be partially or completely restricted, which gives up on some or all of the opportunities for social benefits altogether. The field of *privacy-preserving data analysis* seeks to create a third option by designing methods for analyzing sensitive data that enable researchers to realize the social benefits of the data without compromising individual privacy. However, combining these two goals is a delicate task. Indeed, there have been several high profile "re-identifications" of supposedly anonymous data sets that have lead to tighter control of sensitive data—the re-identification of former Governor of Massachusetts William Weld's medical records [85], re-identification of certain users' movie rental history in the Netflix dataset [69], and the re-identification of users from AOL search records [11] being just a few examples. There are also academic papers suggesting that many more datasets may be re-identifiable, such as databases of genetic information for genome-wide association studies [49], anonymized social network graphs [68], and databases of anonymized writing samples [67].

In this thesis we study privacy-preserving data analysis under *differential privacy*. Differential privacy, introduced by Dwork, McSherry, Nissim, and Smith [30] (building on [26, 34, 15]) is a stringent notion of privacy that guarantees no individual's data has a significant influence on the information released about the database. More precisely, a differentially private algorithm is randomized, and comes with the guarantee that the distribution of the algorithm's outputs remains nearly unchanged, in a precise technical sense, if a single individual's record in the database is added or removed from the dataset. Intuitively, since no individual's data has a significant influence on the output of the algorithm, then it is impossible for the algorithm to reveal a significant amount of information specific to any one individual.[1] We encourage the reader to look at the excellent surveys by Cynthia Dwork [28, 29] for more interpretation of and the motivation behind the definition of differential privacy.

For its strong guarantees, differential privacy comes at a price. Indeed, there has been a great deal of attention on understanding to what extent differential privacy is compatible with *useful* data analysis. In this thesis we will focus on a particular data analysis task—answering *counting queries* on a sensitive database. Counting queries are of the form "What fraction of individual records in the database D satisfy some property q?" In addition to being a natural framework for studying the feasibility of differentially private data analysis, counting queries are a very general primitive that enable a wide variety of computational tasks in statistics, machine learning, and computational learning theory. The line of work on answering counting queries under the constraints of differential privacy has produced some beautiful and striking results, suggesting that a great deal of useful data analysis is possible. These results roughly fall into three categories:

1) Some of the earliest results in differential privacy [26, 15, 30] gave a simple, efficient algorithm capable of privately accurately answering nearly n^2 counting queries, by perturbing the answers to the queries independently with random noise. Here and in the sequel $n = |D|$ denotes the number of individual records in the database.

2) A beautiful and surprising result of Blum, Ligett, and Roth [16] showed how to approximately answer *exponentially many* counting queries on a sensitive database. There have been several exciting developments [32, 46, 78, 35, 44, 42, 43, 50, 70] showing how to achieve

[1] This statement was only intended to provide intuition for why the guarantee of differential privacy has an intuitive relationship to individual privacy. We caution the reader against taking it literally, as it requires care to correctly interpret the guarantees made by differential privacy.

even stronger utility guarantees. All of these algorithms work by perturbing the answers to the queries with carefully correlated noise, and as a result are computationally inefficient.

3) A parallel line of work, beginning with the seminal results of Dinur and Nissim [26] shows that these algorithms achieve nearly optimal utility among all differentially private algorithms in terms of the number of counting queries answered and the desired level of accuracy.

Together, these results give a clear and nearly-complete picture of when accurate data analysis is consistent with differential privacy, at least when we consider counting queries. However, as we've alluded to, some of the most powerful algorithms for differentially private data analysis (those in item 2 above) are not computationally efficient. In many settings, the time to privately answer the queries is exponential in the time to answer the queries without a guarantee of privacy. As a result, the feasibility of differentially private data analysis is far less clear if we simultaneously require differential privacy, accuracy for counting queries, and *computational efficiency*. The issue of computational efficiency is not purely theoretical, as running time considerations have been an important part of several of the empirical papers on answering counting queries under differential privacy [60, 61, 43]. If we cannot achieve all three of these goals simultaneously, at least in practice, then differential privacy cannot resolve the dilemma of analyzing sensitive data.

In addition to computational inefficiency, there are several other issues that threaten to prevent the adoption of differentially private data analysis. One such issue is privacy for the data analysts, which is ignored in the above discussion where we had assumed that the database itself is the only sensitive information we wish to protect. However, the increased value of data also increases the value of proprietary methods for analyzing that data. Unfortunately, due to the need to correlate the answers to different queries in complex ways, answers to queries asked by one analyst of a data set may leak information about the queries asked by another analyst. For example, suppose we have a database consisting of sensitive information about the online activities and purchasing behavior of different consumers. An online retailer may want to query this database in order to train a classifier that will identify products new customers are likely to buy, and in particular this retailer's competitive advantage may be in knowing the best attributes to focus on when training the classifier. If their competitor can also query the database, and the algorithm does not satisfy analyst privacy, then the competing retailer may be able to learn which attributes have been considered by queries most frequently, and thus steal the trade secrets of the first retailer. Revealing this information may

be quite costly for the data analysts. Even if we are not concerned about the economic costs to the online retailer, there is a risk that if we cannot prevent their queries from being leaked, then they may seek other ways to access the data that are less safe for the individuals in the database. In addition to possible economic harms to the analysts, such as in the preceding example, the queries asked to the database may be embarrassing or stigmatizing, for instance if an analyst is trying to determine how many participants in the database would support a political position that is believed to be significantly out of the mainstream. Designing ways to access the database that satisfy differential privacy not only for the data subjects, but also for the data analysts is a potential way to address all of these concerns and ease the path to adoption for privacy-preserving data analysis.

1.1 Contributions of this Thesis

In this thesis we present several new results on the complexity of differentially private data release, and on the possibility of satisfying differential privacy for the analyst. We now give a high-level summary of these results.

1.1.1 The Complexity of Natural Private Data Analysis Tasks

In Chapters 3 and 4 we show that computational complexity is a barrier for effective differentially private data analysis, *even for natural data analysis tasks.*

Our first contribution to this area is to demonstrate that, under standard cryptographic assumptions (namely, the existence of one-way functions), there is no efficient differentially private algorithm capable of accurately answering even $n^{2+o(1)}$ arbitrary counting queries. Here an efficient algorithm would be one whose running time is polynomial in the size of the database and the time required to evaluate the queries without a guarantee of privacy. This result is essentially optimal with respect to the number of queries, as the simplest differentially private algorithm—the Laplace mechanism—accurately answers $\widetilde{\Omega}(n^2)$ counting queries. One way to summarize this result is that if unless we restrict the types of queries we want to answer, or allow exponential running time, then the Laplace mechanism is essentially the best possible differentially private algorithm. We prove this result by refining and extending the connection between differential privacy and cryptographic *traitor-tracing schemes* discovered by Dwork et al. [32]. This work is forthcoming in the Symposium on Theory of Computation (STOC 2013) [88].

Although the previous result is essentially tight with respect to the *number of queries*, it is not tight with respect to the *type of queries*. Indeed, the "hard queries" for the previous result will be rather complex, and unlikely to reflect the type of queries a real data analyst would ask about the database. Thus, a logical question to ask is whether or not it is possible to answer many more than n^2 "simple" queries. Our second contribution is to show even stronger computational barriers that apply even to very simple queries and rule out a particular type of differentially private algorithm. In particular, we show strong hardness results for algorithms that generate *synthetic databases*—roughly, a new database with "fake" records that preserves certain properties of the original database. We show that, assuming the existence of one-way functions, there is no efficient algorithm that generates a private synthetic database even if we only want to preserve the answers to 2-way marginal queries (roughly the means of and pairwise correlations between attributes). Our proof builds on the technique of Dwork et al. [32], who showed a similar result but for a small but seemingly unrealistic family of queries. This work is joint with Salil Vadhan, and appeared in the Theory of Cryptography Conference (TCC 2011) [89].

1.1.2 Faster Algorithms for Privately Answering Marginal Queries

A natural approach to circumventing the computational hardness results of Chapters 3 and 4 is to find algorithms capable of answering large numbers of simple, natural counting queries, without generating synthetic data. Arguably the simplest, natural family of queries are *k-way marginal queries*. A k-way marginal query is asked on a database D whose records are strings of d bits, specifying each individual's value for d binary attributes. The query is specified by a set $S \subseteq [d]$, $|S| \leq k$, and a pattern $t \in \{0,1\}^{|S|}$. The query asks, "What fraction of the individual records in D has each of the attributes $j \in S$ set to t_j?" The set of k-way marginal queries, also known as the "k-way contingency table" of the database, is a workhorse for statistical data analysis at organizations such as the US Census Bureau, and are a natural test-case for the power and feasibility of differentially private data analysis. The number of k-way marginal queries is roughly $d^{\Theta(k)}$, thus we could answer all such queries in time $d^{\Theta(k)}$ on a database of size $d^{\Theta(k)}$ using the Laplace mechanism.

A previous series of results [41, 22, 45, 38] have used techniques from *computational learning theory* to circumvent the computational barriers and give more efficient algorithms for privately releasing marginal queries. However, these results leave several open questions that we partially

address. One weakness of these algorithms is that they do not yield accurate answers to every marginal query, but rather give answers that have small average error over various distributions on marginal queries. In Chapter 5, we give a differentially private algorithm that releases a summary of the database from which an analyst can compute an accurate answer to *every* k-way marginal query. The running time of the algorithm and the summary is $d^{O(\sqrt{k})}$ and the answers will be accurate as long as $n \geq d^{O(\sqrt{k})}$. This work is joint with Justin Thaler and Salil Vadhan, and appeared in the International Colloquium on Automata, Languages, and Programming (ICALP 2012) [87].

In order to give accurate answers, the algorithm in Chapter 5 and all prior algorithms for this problem require a database that is much larger than what would be needed for an exponential time algorithm. Recent experimental results by Hardt et al. [43] suggest that an exponential-time algorithm based on the private multiplicative weights mechanism [44] is not far from being practical and provides much better accuracy than other techniques when the database is relatively small. Motivated by their experiments, we consider the possibility of designing algorithms for answering k-way marginal queries on "small" databases with subexponential running time. In Chapter 6, we give a differentially private algorithms that approximately answers all k-way marginal queries on a database of size $\Theta(kd^{.51})$ in time $\exp(d^{1-\Omega(1/\sqrt{k})})$. The minimum database size required by our algorithm nearly matches the best-known information theoretic upper bound of $\widetilde{O}(k\sqrt{d})$ [44] and matches the best-known information theoretic lower bound of $\widetilde{\Omega}(\max\{k, \sqrt{d}\})$ when k is small [76, 90].

The algorithms of Chapter 5 and 6 are all based on the idea of encoding the answers to the k-way marginal queries in a function (depending on the sensitive database) and privately learning that function, which was introduced by Gupta et al. [41].[2] Our private learning algorithms are based on approximations to that function by low-degree polynomials, and in Chapter 6, we prove various new upper and lower bounds on the degree of such approximating polynomials that may have applications outside of differential privacy. The results of Chapter 6 are joint with Karthik Chandrasekaran, Justin Thaler, and Andrew Wan [21].

[2]Here, the notion of private learning is defined with respect to the database defining the function being learned, in contrast to previous works on "private learning" (e.g. [51]) where privacy is defined with respect to the examples given to the learning algorithm.

1.1.3 Differential Privacy for the Analyst

Computational complexity is not the only barrier to effective differentially private data analysis. Another barrier is that differentially private algorithms may not be entirely "safe" for an analyst to use, as essentially any algorithm capable of answering large number of queries must correlate its answers to the queries, and thus leak information about the sequence of queries that were asked [33]. While this leakage seems harmless if we assume the queries are issued by a single data analyst, in practice there will be multiple parties interested in analyzing sensitive data sets, and the queries they ask may themselves be sensitive or proprietary. Motivated by these concerns, Dwork, Naor, and Vadhan [33] introduced the notion of *analyst differential privacy*, and showed the existence of an algorithm that answers exponentially many queries on a sensitive database and ensures differential privacy not only for the database but for the queries made on the database.

The algorithm of Dwork et al. [33] suffers a few shortcomings: Their algorithm does not achieve an optimal rate of error compared with algorithms that do not guarantee analyst privacy. Also, their algorithm does not ensure differential privacy for the analyst if the other analysts collude or make queries under multiple identities. Finally, their algorithm is only capable of answering counting queries, and not arbitrary low-sensitivity queries.

In Chapter 7 we present a suite of new analyst-private algorithms. Some of our algorithms answer linear queries with error $\mathrm{poly}(d, |Q|)/\sqrt{n}$, and thus has the optimal dependence on n, even among algorithms that do not guarantee analyst privacy [26]. Another algorithm is capable of answering exponentially many counting queries while guaranteeing differential privacy for the entire set of queries asked by a single analyst, even if all other analysts collude. Yet another algorithm is capable of answering arbitrary low-sensitivity queries. All of our algorithms are inspired by a novel view of the private counting query release problem as privately computing equilibrium strategies for a two-player zero-sum game. This viewpoint unifies and generalizes several previous approaches, in addition to enabling new results on analyst privacy.

Chapter 2

Background

In this chapter we will introduce the notation and standard results about differential privacy that we rely on throughout this thesis.

2.1 Databases

We define a *database* $D \in (\{0,1\}^d)^n$ to be a collection of n rows $(x^{(1)}, \ldots, x^{(n)}) \in \{0,1\}^d$. We will always assume that n is public information (and thus not subject to privacy considerations) and that the order of rows in D is irrelevant. We sometimes refer to $\{0,1\}^d$ as the *data universe*. Two databases $D, D' \in (\{0,1\}^d)^n$ are *adjacent* if they differ only on a single row, and we denote this relationship by $D \sim D'$. An alternative way to represent the database is not as a collection of rows from $\{0,1\}^d$, but rather as a probability distribution over $\{0,1\}^d$. In this representation, we view the database as a probability mass function $D\colon \{0,1\}^d \to [0,1]$ where $D(x)$ is the fraction of rows in D equal to the universe element x. For any two adjacent databases $D \sim D'$, the resulting probability distributions will satisfy

$$\frac{1}{2} \sum_{x \in \{0,1\}^d} |D(x) - D(x')| = \frac{1}{n}.$$

2.2 Differential Privacy

The notion of differential privacy was formalized in the work of Dwork, McSherry, Nissim, and Smith [30].

Definition 2.1 (Differential Privacy)**.** Let $\mathcal{M}\colon (\{0,1\}^d)^n \to \mathcal{R}$ be a randomized algorithm that takes a database as input (where n and d are varying parameters). $\mathcal{M}$ is (ε, δ)-*differentially private* if for every two adjacent databases $D \sim D'$ and every $S \subseteq \mathcal{R}$,

$$\Pr\left[\mathcal{M}(D) \in S\right] \leq e^{\varepsilon}\Pr\left[\mathcal{M}(D') \in S\right] + \delta.$$

If $\mathcal{M}$ is (ε, δ)-differentially private for some functions $\varepsilon = \varepsilon(n) = O(1)$, $\delta = \delta(n) = \mathrm{negl}(n)$, we will drop the parameters ε and δ and say that $\mathcal{M}$ is *differentially private*.

Informally, differential privacy ensures that the contents of a single row of the database do not significantly influence the output distribution of the algorithm. As a consequence, an adversary who knows all the rows of the database but one, "cannot learn much" from seeing the output of a differentially private algorithm applied to that database.

Differential privacy is easy to achieve on its own; simply ignore the database and output $\perp$. Thus, the goal is to find useful algorithms that satisfy this definition. For example, it is possible to approximately compute the value of a function on the database while satisfying differential privacy by evaluating that function, and then adding noise whose standard deviation is somewhat larger than the maximum influence a single row of the database can have on the output of the function. By doing so, the contents of that row do not significantly influence the output. This approach is typically known as the Laplace mechanism, and is discussed in more detail in Section 2.4.1.

In this thesis, we will assume that the range $\mathcal{R}$ is discrete. Doing so has a number of advantages: it avoids subtleties in the definition of differential privacy that arise when the range is continuous and it also avoids certain security vulnerabilities that arise in the implementation of real-valued differentially private algorithms using floating-point arithmetic, discovered in a clever attack by Mironov [65]. Nonetheless, many differentially private algorithms, such as the Laplace mechanism, are mathematically cleaner to define and analyze using a continuous range of outputs. In Section 2.4.1 we discuss how to implement the continuous Laplace mechanism using a discrete range without significantly degrading its guarantees. For the other algorithms we discuss, we will suppress this issue and use a continuous range when appropriate, with the understanding that these algorithms can be modified to have a discrete range without affecting the results substantively.

We conclude this section with a useful lemma that gives a sufficient condition for an algorithm to be differentially private. In some cases it is easier to establish this condition than it is to establish differential privacy directly.

Lemma 2.2 ([35]). *Let $\mathcal{M}\colon (\{0,1\}^d)^n \to \mathcal{R}$ be an algorithm. For any $\varepsilon, \delta > 0$, if it holds that for every pair of adjacent databases $D \sim D'$,*

$$\Pr_{r \leftarrow_R \mathcal{M}(D)}\left[\left|\ln\left(\frac{\Pr\left[\mathcal{M}(D)=r\right]}{\Pr\left[\mathcal{M}(D')=r\right]}\right)\right| \leq \varepsilon\right] \geq 1-\delta,$$

then $\mathcal{M}$ satisfies (ε, δ)-differential privacy.

Note that, since we have assumed $\mathcal{R}$ is discrete, we are justified in writing $\Pr\left[\mathcal{M}(D)=r\right]$.

2.2.1 Composition of Differentially Private Algorithms

An important property of differential privacy is that it is well-behaved under composition. The simplest composition theorem for differential private states that if $\mathcal{M}_1$ and $\mathcal{M}_2$ are (ε, δ)-differentially private algorithms then the algorithm $\mathcal{M}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$ is a $(2\varepsilon, 2\delta)$-differentially private algorithm. By induction, this property implies that if we apply T (ε, δ)-differentially private algorithms to D, then the result is $(T\varepsilon, T\delta)$-differentially private. For our results, we will use an even stronger composition property, due to Dwork, Roth, and Vadhan [35]. First, they showed that the rate at which differential privacy decays under T-fold composition can actually be shown to be approximately $\sqrt{T}$, rather than linear in T. They also showed that these bounds on the deterioration of privacy under composition hold even if the sequence of algorithms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ is chosen *adaptively*. We model adaptive composition via a game with a (computationally unbounded) privacy adversary $\mathcal{A}$, described in Figure 1.

Let $D \in (\{0,1\}^d)^n$ be a database, $\mathcal{A}$ be an adversary, and $T \in \mathbb{N}$ be a parameter.
For $t = 1, \ldots, T$**:**
$\mathcal{A}(y_1, \ldots, y_{t-1})$ chooses an $(\varepsilon_0, \delta_0)$-differentially private algorithm $\mathcal{M}_t\colon (\{0,1\}^d)^n \to \mathcal{R}$.
$y_t \leftarrow_{\mathrm{R}} \mathcal{M}_t(D)$.
End For
Output: $y_1, \ldots, y_T$

Figure 1: A framework for adaptive composition.

We can now formally state the composition properties of differential privacy as follows:

Theorem 2.3 (Composition of Differential Privacy [30, 35])**.** *Fix any $1/2 \geq \varepsilon_0, \delta_0 > 0$. For every adversary $\mathcal{A}$, database $D \in (\{0,1\}^d)^n$, and $y_1, \ldots, y_T \in \mathcal{R}$, let $\mathcal{M}_{\mathcal{A}}(D)$ be the algorithm obtained by performing the experiment of Figure 1 and outputting $y_1, \ldots, y_T$. Then we have*

1) $\mathcal{M}_{\mathcal{A}}$ is $(T\varepsilon_0, T\delta_0)$-differentially private, and

2) for any $\delta > 0$, $\mathcal{M}_{\mathcal{A}}$ is $(\varepsilon, T\delta_0 + \delta)$-differentially private for

$$\varepsilon = \sqrt{2T\log(1/\delta)}\varepsilon_0 + 2T\varepsilon_0^2.$$

In particular, if an algorithm $\mathcal{M}$ can be expressed as the T-fold adaptive composition of $(\varepsilon_0, 0)$-differentially private algorithms for $\varepsilon_0 \leq \varepsilon/T$, then $\mathcal{M}$ is $(\varepsilon, 0)$-differentially private. Further, if $\mathcal{M}$ can be expressed as the T-fold adaptive composition of $(\varepsilon_0, 0)$-differentially private algorithms for $\varepsilon_0 \leq \varepsilon/\sqrt{8T\log(1/\delta)}$, then $\mathcal{M}$ is (ε, δ)-differentially private.

2.3 Sanitizers and Counting Queries

For our study, we must define a data analysis task that we would like to accomplish while satisfying differential privacy. In this work we focus on differentially private algorithms that answer *counting queries* on the sensitive database. A counting query on $\{0,1\}^d$ is defined by a function $q\colon \{0,1\}^d \to [0,1]$. Counting queries are also referred to as *linear queries*.[3] Abusing notation, we define the evaluation of a query q on a database $D = (x^{(1)}, \ldots, x^{(n)}) \in (\{0,1\}^d)^n$ to be

$$q(D) = \frac{1}{n}\sum_{i=1}^{n} q(x^{(i)})$$

When d is a varying parameter, we use $\mathcal{Q}^{(d)}$ to denote a set of counting queries on $\{0,1\}^d$ and $\mathcal{Q} = \bigcup_{d\in\mathbb{N}} \mathcal{Q}^{(d)}$. Typically we simply write $\mathcal{Q}$ when d is clear from context.

We call algorithms that answer counting queries *sanitizers*. A sanitizer takes a database and a sequence of counting queries from some family $\mathcal{Q}$ as input and returns a sequence of answers to those queries. Formally a sanitizer is a function $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathcal{Q}^{(d)})^k \to \mathbb{R}^k$ (where n, d, and k are varying parameters). Here we assume that $\mathcal{M}$ simply returns a list of k real-valued answers, with the understanding that the j-th component of the output is an answer to the j-th query.

[3]Typically a counting query is define by a predicate $q\colon \{0,1\}^d \to \{0,1\}$ whereas linear query is defined by a real-valued function $q\colon \{0,1\}^d \to [0,1]$. However, we use counting query for the more general definition.

This assumption will turn out to be without loss of generality, even if we consider computational efficiency. Any algorithm that encodes the answers to each of the queries in some arbitrary efficient data structure can be converted to one that encodes its answers as a list of real numbers by querying the data structure k times. Since the length of $\mathcal{M}$'s input is at least k, this additional computation time is acceptable. Definition 2.1 extends naturally to sanitizers by requiring that for every $Q = (q_1, \ldots, q_k) \subseteq \mathcal{Q}$, the sanitizer $\mathcal{M}_Q(\cdot) = \mathcal{M}(\cdot, Q)$ is (ε, δ)-differentially private as a function of the input database.

Now we formally define what it means for a sanitizer to accurately answer counting queries

Definition 2.4 (Accuracy of Sanitizers)**.** Let D be a database and $Q = (q_1, \ldots, q_k)$ be a set of counting queries and $\alpha > 0$ be a parameter. A sequence of answers $a_1, \ldots, a_k$ is *α-accurate for $q_1, \ldots, q_k$ on D* if

$$\forall j \in [k], |q_j(D) - a_j| \leq \alpha.$$

Let $\mathcal{Q}$ be a set of counting queries, $k \in \mathbb{N}$ and $\alpha, \beta > 0$ be parameters. A sanitizer $\mathcal{M}$ is (α, β)-*accurate for k queries from $\mathcal{Q}$* if for every database $D \in (\{0,1\}^d)^n$ and every sequence of queries $Q = (q_1, \ldots, q_k) \subseteq \mathcal{Q}$

$$\Pr_{\mathcal{M}\text{'s coins}}[\mathcal{M}(D, Q) \text{ is } \alpha\text{-accurate for } Q \text{ on } D] \geq 1 - \beta.$$

In some cases we will suppress certain parameters from the definition of accuracy. In particular, if $\mathcal{M}$ is (α, β)-accurate for k queries from $\mathcal{Q}$, for every $k \in \mathbb{N}$ and some α, β (possibly depending on the other parameters), we will say that $\mathcal{M}$ is (α, β)-*accurate for $\mathcal{Q}$*. When $\mathcal{Q}$ is clear from context we will suppress it, and if β is omitted it should be assumed to be some function $\beta = \beta(n) = \mathrm{negl}(n)$. If $\mathcal{M}$ is α-accurate for k queries from $\mathcal{Q}$, for $\alpha = 1/3$, we will drop α and β and say that $\mathcal{M}$ is *accurate for k queries from $\mathcal{Q}$*.

The above definition of a sanitizer captures the setting where the queries are specified by the analyst. Many applications of differential privacy necessitate a *one-shot data release*, where the data owner computes and publishes a single differentially private summary of the database that enables analysts to compute an answer to any query in $\mathcal{Q}$. In this setting, we typically think of the class of queries as fixed, rather than being input to the sanitizer. We call such a sanitizer a *one-shot sanitizer*. Conceptually, the difference between the two models is that a sanitizer is answering a set Q of queries explicitly requested by the data analyst, and thus it is both natural and necessary

to run in time $\mathrm{poly}(|Q|)$. However, in the one-shot model, we typically imagine that the family $\mathcal{Q}$ of queries contains all the queries the analyst may ask, which is a much larger set that those the analyst will actually compute. Thus in the one-shot model the natural running time for the algorithm may even be sublinear in $|\mathcal{Q}|$ (e.g. when $\mathcal{Q}$ contains all marginal queries, and $|\mathcal{Q}| = 2^d$). We will consider the one-shot model in Chapters 4-6 when we study the family of marginal queries.

When concerned about computational complexity, it is not without loss of generality to assume that a one-shot sanitizer outputs a list of $|\mathcal{Q}|$ real-valued answers, since $|\mathcal{Q}|$ may be exponentially large in the length of the sanitizer's input. Thus, we allow $\mathcal{M}$ to output an arbitrary data structure S from a range $\mathcal{S}$, and that there be an *evaluator* function $\mathcal{E} : \mathcal{S} \times \mathcal{Q} \to \mathbb{R}$ that estimates $q(D)$ from the output of $\mathcal{M}(D)$ and the description of q. For example, $\mathcal{M}$ may output a vector $S = (q(D) + Z_q)_{q \in \mathcal{Q}}$ where Z_q is a random variable for each $q \in \mathcal{Q}$, and $\mathcal{E}(S, q)$ is the q-th component of $S \in \mathcal{R} = \mathbb{R}^{|\mathcal{Q}|}$. Abusing notation, we will write $q(S)$ and $q(\mathcal{M}(D))$ as shorthand for $\mathcal{E}(S, q)$ and $\mathcal{E}(\mathcal{M}(D), q)$, respectively.

We will say that a one-shot sanitizer $\mathcal{M}$ is accurate for the concept class $\mathcal{Q}$ if the answers $q(\mathcal{M}(D))$ are close to the fractional counts $q(D)$. Formally

Definition 2.5 (Accuracy for One-Shot Sanitizers)**.** An output S of a one-shot sanitizer $\mathcal{M}(D)$ is α-*accurate* for a family of counting queries $\mathcal{Q}$ on D if

$$\forall q \in \mathcal{Q}, |q(S) - q(D)| \leq \alpha.$$

A one-shot sanitizer $\mathcal{M}$ is (α, β)-*accurate for* $\mathcal{Q}$ if for every database D,

$$\Pr_{\mathcal{M}\text{'s coins}}[\mathcal{M}(D) \text{ is } \alpha\text{-accurate for } \mathcal{Q} \text{ on } D] \geq 1 - \beta.$$

When $\mathcal{Q}$ is clear from context, we simply write that $\mathcal{M}$ is an (α, β)-accurate one-shot sanitizer.

Synthetic Data. It is especially desirable to design one-shot sanitizers that generate *synthetic data*. A one-shot sanitizer generates synthetic data if its output is a summary $\widehat{D} \in (\{0,1\}^d)^{\widehat{n}}$, for a suitable choice of $\widehat{n}$, and its evaluator is $\mathcal{E}(\widehat{D}, q) = q(\widehat{D})$. That is, the output is a new database with the same number of columns (though possibly a different number of rows), and the evaluation of a query is performed simply by evaluating the query on this new database as if it were the original database. In Chapter 4 we show strong computational hardness results for one-shot sanitizers that generate private synthetic data.

2.3.1 Efficiency of Sanitizers

Simply stated, we view a sanitizer as efficient if it runs in time polynomial in the length of its input. For a one-shot sanitizer, the only input is the database $D \in (\{0,1\}^d)^n$, thus an *efficient one-shot sanitizer* runs in time $\mathrm{poly}(d, n)$. In Chapters 4-6 we will take a more refined view and show specific upper and lower bounds on the time complexity of certain one-shot sanitizers, but it is useful to keep in mind that $\mathrm{poly}(d, n)$ is the natural notion of efficiency for a one-shot sanitizer.

For sanitizers that are not one-shot, the natural notion of efficiency will depend on how the queries are given to the sanitizer as input. Notice that to specify an arbitrary counting query $q\colon \{0,1\}^d \to \{0,1\}$ requires 2^d bits. In this case, a sanitizer whose running time is polynomial in the time required to specify the query is not especially efficient. Thus, we restrict attention to queries that are efficiently computable, so are not the bottleneck in computation. For our results, we will fix the representation to be Boolean circuits over the basis $\{\wedge, \vee, \neg\}$ with possibly unbounded fan-in. In this representation, any query can be evaluated in time $|q|$, where $|\cdot|$ denotes the size of the circuit computing q. We also want to consider the case where the queries are computable by circuits of low depth. For a constant $h \in \mathbb{N}$, we use $\mathcal{Q}^{(d)}_{\mathsf{depth}-h}$ to denote the set of all counting queries on $\{0,1\}^d$ specified by circuits of depth h. Finally, we use $\mathcal{Q}^{(d)}_{\mathsf{all}}$ to denote the set of all counting queries on $\{0,1\}^d$.

Definition 2.6 (Efficiency of Sanitizers)**.** A sanitizer $\mathcal{M}$ is *efficient* if, on input a database $D \in (\{0,1\}^d)^n$ and k queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}_{\mathsf{all}}$, $\mathcal{M}$ runs in time $\mathrm{poly}(d, n, |q_1| + \cdots + |q_k|)$. For every $h \in \mathbb{N}$, a sanitizer $\mathcal{M}$ is *efficient for depth-h* queries if, on input a database $D \in (\{0,1\}^d)^n$ and k queries $q_1, \ldots, q_k \in \mathcal{Q}^{(d)}_{\mathsf{depth}-h}$, $\mathcal{M}$ runs in time $\mathrm{poly}(d, n, |q_1| + \cdots + |q_k|)$.

2.4 Some Differentially Private Algorithms

We now introduce two interesting differentially private algorithms. In this chapter we focus on the algorithms necessary to provide context for our new results. In later chapters we will introduce additional algorithmic techniques as they are needed.

2.4.1 The Laplace Mechanism

The most basic differentially private algorithm is the so-called *Laplace mechanism*, introduced by Dwork, McSherry, Nissim, and Smith [30]. Essentially, the Laplace mechanism computes the correct answers to the queries and perturbs them independently with noise from a Laplace distribution. We can apply the Laplace mechanism to more than just counting queries; it can be applied more generally to *low-sensitivity queries*. Let $q\colon (\{0,1\}^d)^n \to \mathbb{R}$ be a real-valued function. We define the *(global) sensitivity* of q as

$$\Delta_q = \max_{D \sim D'} |q(D) - q(D')| .$$

If q has sensitivity Δ, then we say q is Δ-sensitive. For a parameter $\sigma > 0$, we define the *Laplace distribution*, $\mathrm{Lap}(\sigma)$ over $\mathbb{R}$ to be have the probability density function $\mathrm{Lap}(\sigma)[z] \propto \exp(-|z|/\sigma)$.

Definition 2.7 (Laplace mechanism [30])**.** Let $\mathcal{M}_{\mathrm{Lap}(\sigma)}(D, q) = q(D) + Z$ where Z is chosen according to $\mathrm{Lap}(\sigma)$.

The definition of accuracy (Definitions 2.4 and 2.5) extends naturally to answering real-valued queries other than counting queries.

For an appropriate choice of σ, the Laplace mechanism guarantees differential privacy.

Theorem 2.8 ([30])**.** *Let q be a query with global sensitivity Δ_q. For $\sigma = \Delta_q/\varepsilon$, the Laplace mechanism $\mathcal{M}_{\mathrm{Lap}(\sigma)}(D, q)$ is (ε, δ)-differentially private.*

An elementary fact about the Laplace distribution is that for any $\sigma, t > 0$,

$$\Pr_{Z \leftarrow_{\mathrm{R}} \mathrm{Lap}(\sigma)} [|Z| > t\sigma] \leq e^{-t}.$$

This fact will be useful for bounding the error introduced by the Laplace mechanism in order to obtain privacy.

When answering a small number of queries with low-sensitivity, the Laplace mechanism provides good accuracy. In particular, we can obtain the following lemma, which follows by combining Theorem 2.8, composition theorems for differential privacy (Theorem 2.3) and the above fact about the tails of the Laplace distribution.

Lemma 2.9. *Let $Q = \{q_1, \ldots, q_k\}$ be a set of Δ-sensitive queries $q_j\colon (\{0,1\}^d)^n \to \mathbb{R}$, and let $D \in (\{0,1\}^d)^n$ be a database. For any $\sigma > 0$, let $\mathcal{M}_{\mathrm{Lap}(\sigma)}(D, Q)$ be the sanitizer that outputs*

$\mathcal{M}_{\mathrm{Lap}(\sigma)}(D, q_j)$ for each $q_j \in Q$ (using independent randomness for each query). Then the following both hold:

1) *For every $\varepsilon, \beta > 0$, if $\sigma = \Delta k/\varepsilon$, then $\mathcal{M}_{\mathrm{Lap}(\sigma)}$ is $(\varepsilon, 0)$-differentially private and (α, β)-accurate for*
$$\alpha = \frac{\Delta k \log(k/\beta)}{\varepsilon}.$$

2) *For every $\varepsilon, \delta, \beta > 0$ if $\sigma = \Delta\sqrt{8k\log(1/\delta)}/\varepsilon$, then $\mathcal{M}_{\mathrm{Lap}(\sigma)}$ is (ε, δ)-differentially private and (α, β)-accurate for*
$$\alpha = \frac{\Delta\sqrt{8k\log(1/\delta)}\log(k/\beta)}{\varepsilon}.$$

The Discrete Laplace Mechanism

As we discussed, we will actually assume that algorithms have discrete outputs. In addition to simplifying the definition of privacy, assuming a discrete range will also make it easier to make a precise statement about the complexity of the Laplace mechanism. Although we will continue view the output of the Laplace mechanism as a real number, since this leads to cleaner mathematical statements, for purposes of computational complexity we will assume that the output of the Laplace mechanism is rounded to a discrete range. That is, to compute the answer, we will round the true answer $q(D)$ down to an integer multiple of τ, for a suitable choice of $\tau > 0$, and then add a noise term $Z \in \tau\mathbb{Z}$ from a discretized version of the Laplace mechanism $\mathrm{Lap}_\tau(\sigma)$, in which $\Pr_{Z \leftarrow_R \mathrm{Lap}_\tau(\sigma)}[Z = z] \propto \exp(-z/\sigma)$ for every $z \in \tau\mathbb{Z}$.

First, we observe that rounding can only increase the sensitivity of a query by τ. We will choose τ small enough that $\Delta + \tau \leq 2\Delta$, and the discretized Laplace mechanism remains 2ε-differentially private. In most cases (e.g. counting queries) we could even eliminate this constant-factor blowup. Since all of the results of this paper are asymptotic, we will ignore this constant-factor increase. Similarly, privacy is not affected when we apply the Laplace mechanism to multiple queries and apply the composition theorem.

Second, we note that $\mathrm{Lap}_\tau(\sigma)$ can be sampled in expected time $\approx e^{-\tau/\sigma}$ using the following procedure: generate a random sign $b \in \{0, 1\}$. Then, generate a non-negative integer z by initializing z to 0 and then repeatedly incrementing z by 1, deciding to stop after each increment independently with probability $1 - e^{-\tau/\sigma}$. Finally, output $Z = b \cdot z$. If we want a worst-case bound on running time we can stop the incrementing of z at a suitable value and this will come at

the expense of achieving only (ε, δ)-differential privacy for $\delta > 0$. Since the appropriate value at which to stop will depend on δ, and in turn on the application, we will omit the details here.

Finally, we need to consider how τ should be set. In all of our applications, it will be sufficient to take $\tau = \mathrm{poly}(\sigma)$ (note that in most applications, e.g. counting queries, $\sigma < 1$). Since discretizing to multiples of τ can only introduce additional error τ, and we are already accounting for error at least $\sigma \gg \tau$, discretizing will only introduce additional error that is smaller than a constant factor. Again, since all of our accuracy bounds are asymptotic, we will simply ignore this constant factor when proving accuracy, and ignore the small additional error incurred by discretization.

Summary of the Laplace Mechanism

The only additional work done by the Laplace mechanism beyond what would be required to answer the queries non-privately is to add independent noise to each query. As we discussed, for a discrete version of the Laplace mechanism this noise can be added in time $\mathrm{poly}(\sigma)$ per query. Thus the mechanism is clearly efficient in the sense of Definition 2.6. Also note that for a counting query, $q(D) = \frac{1}{n}\sum_{i=1}^{n} q(x^{(i)})$, the sensitivity is $\Delta = 1/n$. So we can summarize the properties of the Laplace mechanism for answering counting queries with the following corollary:

Corollary 2.10. *For an appropriate choice of σ, the Laplace mechanism $\mathcal{M}_{\mathrm{Lap}}$ is 1) differentially private, 2) efficient, and 3) accurate for $\widetilde{\Omega}(n^2)$ queries from $\mathcal{Q}_{\mathsf{all}}$.*

We remark that this statement is essentially tight, because there exists a $k = \widetilde{O}(n^2)$ such that for any choice of $\sigma > 0$, the Laplace mechanism does not satisfy both differential privacy and non-trivial accuracy for k counting queries.

2.4.2 The Private Sparse Vector Technique

An extremely useful building block for differentially private sanitizers is the *private sparse vector algorithm*. This technique was first introduced in the work of Dwork et al. [31] and has subsequently been abstracted and generalized (cf. [77]). Intuitively the sparse vector algorithm takes as input a database and a set of low-sensitivity queries, with the promise that only a small number of the queries have large answers on the input database. Its output is a set of queries with large answers on the input database. For this work, we will need a slight modification of the private sparse vector algorithm that ensures $(\varepsilon, 0)$-one-query-to-many-analyst differential privacy (here we

assume each query has been issued by a separate analyst). We achieve this additional property by randomizing the parameter $\widehat{\ell}$, specifying the number of queries with large answer that are allowed before the sparse vector algorithm terminates. This modification is the only place in which our presentation of the algorithm differs from the standard presentation (e.g. [77]).

$\mathcal{M}_{\mathrm{SV}}(D, Q = \{q_1, \ldots, q_k\}, \alpha, \beta, \ell)$:
  Let $\varepsilon_0 = \varepsilon/\sqrt{8\ell \log(1/\delta)}$, $\sigma = 2\Delta/\varepsilon_0$, $c = 0$.
  Let $\widehat{\ell} = 2\ell + z$ where $z \leftarrow_{\mathrm{R}} \mathrm{Lap}(1/\varepsilon)$
  For: $j = 1, \ldots, k$
    Let $\widehat{\alpha}_j = \alpha + 2\sigma \log(4k/\beta) + v_j$ where $v_j \leftarrow_{\mathrm{R}} \mathrm{Lap}(\sigma)$.
    Let $z_j \leftarrow_{\mathrm{R}} \mathrm{Lap}(\sigma)$.
    If: $q_j(D) + z_j \geq \widehat{\alpha}_j$ **Then:**
      Output: $q_j(D) + z_j$
      Let $c = c + 1$.
    Else:
      Output: $\perp$
    End If.
    If: $c \geq \widehat{\ell}$ **Then: Halt.**
  End For.

Figure 2: The private sparse vector algorithm.

We can summarize the properties of the private sparse vector algorithm (Figure 2) in the following lemma.

Lemma 2.11 (Modification of [77])**.** *Fix $\varepsilon, \delta > 0$ in the private sparse vector algorithm. Let $Q = \{q_1, \ldots, q_k\}$ be a set of Δ-sensitive queries, $q_j \colon (\{0,1\}^d)^n \to \mathbb{R}$. Let $D \in (\{0,1\}^d)^n$ be a database. Let $\alpha, \beta \in \mathbb{R}$ and $\ell \in \{1, \ldots, k\}$ be such that $\ell \geq \log(4/\beta)/\varepsilon$ and*

$$|\{j \mid q_j(D) > \alpha\}| \leq \ell.$$

Then the following all hold:

1) *$\mathcal{M}_{\mathrm{SV}}$ is (ε, δ)-differentially private.*

2) *$\mathcal{M}_{\mathrm{SV}}(D, Q, \alpha, \beta, \ell)$ returns a set $I \subseteq \{1, \ldots, k\}$ such that with probability at least $1 - \beta$, I is of size at most 3ℓ and contains the index of every query with answer significantly larger than α. Specifically, with probability at least $1 - \beta$,*

$$\left\{ j \mid q_j(D) \geq \alpha + \frac{24\Delta\sqrt{\ell \log(1/\delta)} \log(4k/\beta)}{\varepsilon} \right\} \subseteq I$$

3) *Finally, for every $j \in \{1, \ldots, k\}$, the j-th output does not depend significantly on any other query. Specifically if $Q' = \left\{q_1, \ldots, q_{j-1}, q'_{j'}, q_{j+1}, \ldots, q_k\right\}$, then for every D, every $j \neq j'$, and every $I \subseteq \{1, \ldots, k\} \setminus \{j'\}$*

$$\Pr\left[\mathcal{M}_{\mathrm{SV}}(D, Q) = I\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}_{\mathrm{SV}}(D, Q') = I\right].$$

The first two properties articulated in Lemma 2.11 are reasonably intuitive and describe the privacy and utility properties of the private sparse vector algorithm. Property 3 is somewhat non-standard, and states that the j-th output of the sparse vector algorithm is not significantly influenced by the j'-th query for $j' \neq j$. In other words, the sparse vector algorithm satisfies differential privacy for a single input query. The intuition for why Property 3 holds is that all of the outputs other than the j'-th output can be simulated using only the queries $Q \setminus \{q_{j'}\}$ and the current value of $\widehat{\ell} - c$, and only the latter quantity contains any information about $q_{j'}$—specifically, which branch of the if-statement $q_{j'}$ fell into can affect this quantity by at most 1. However, since noise chosen from $\mathrm{Lap}(1/\varepsilon)$ was added to $\widehat{\ell}$, the differential privacy of the Laplace mechanism ensures that this change of at most 1 is largely hidden. Finally, we remark that the queries used in the sparse vector algorithm need not be specified ahead of time, as each query is processed individually.

We will rely on the sparse vector technique itself in Chapter 7 as a tool for constructing sanitizers that satisfy analyst differential privacy.

2.4.3 The IDC Framework

One of the many powerful applications of the sparse vector technique is to convert a certain kind of "online learning algorithm" called an *iterative database construction (IDC)* into a differentially private sanitizer. This framework for designing differentially private sanitizers—the "IDC framework"—was introduced by the author in joint work with Gupta and Roth [42], although we

have chosen not to emphasize the results in this thesis. In this section we will introduce the definition of an IDC and discuss how it can be used to construct sanitizers.

Roughly, an IDC works by maintaining a sequence of data structures $S^{(1)}, S^{(2)}, \cdots \in \mathcal{S}$ that give increasingly good approximations to some database D with respect to answering queries from a family $\mathcal{Q}$. Moreover, the mechanism produces the next approximation in the sequence by considering only one query $y^{(t)}$ that "distinguishes" the real database in the sense that

$$|q^{(t)}(S^{(t)}) - q^{(t)}(D)|$$

is large.[4]

Syntactically, an IDC is an object of the following type. There is an *update function* $\mathbf{U} : \mathcal{S} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{S}$. The inputs to $\mathbf{U}$ are a *database summary* $S \in \mathcal{S}$, which represents the current approximation; a query $q \in \mathcal{Q}$, which *distinguishes* the approximation S from the true database D; and also a real number that estimates $q(D)$. Formally, we define a *database update sequence* to capture the sequence of inputs to $\mathbf{U}$ used to generate the sequence $S^{(1)}, S^{(2)}, \ldots$.

Definition 2.12 (Database Update Sequence)**.** Let $D \in (\{0,1\}^d)^n$ be any database and let $\left\{(S^{(t)}, q^{(t)}, \widehat{a}^{(t)})\right\}_{t=1,\ldots,C} \in (\mathcal{S} \times \mathcal{Q} \times \mathbb{R})^C$ be a sequence of tuples. We say the sequence of updates is an $(\mathbf{U}, D, \mathcal{Q}, \alpha, C)$-*database update sequence* if the following properties are all satisfied:

1) $S^{(1)} = \mathbf{U}(\emptyset, \cdot, \cdot)$.

2) For every $t = 1, 2, \ldots, C$, $\left|q^{(t)}(D) - q^{(t)}(S^{(t)})\right| \geq \alpha$.

3) For every $t = 1, 2, \ldots, C$, $\left|q^{(t)}(D) - \widehat{a}^{(t)}\right| \leq \alpha/2$.

4) For every $t = 1, 2, \ldots, C-1$, $S^{(t+1)} = \mathbf{U}(S^{(t)}, q^{(t)}, \widehat{a}^{(t)})$.

We note that for all of the iterative database constructions we consider, the approximate answer $\widehat{a}^{(t)}$ is used only to determine the *sign* of $q^{(t)}(D) - q^{(t)}(S^{(t)})$, which is the motivation for requiring that $\widehat{a}^{(t)}$ have error smaller than α. The main measures of efficiency we're interested in from an iterative database construction are the maximum number of updates we need to perform before the database $S^{(t)}$ approximates D well with respect to the queries in $\mathcal{Q}$ and the time required to compute $\mathbf{U}$. To this end we define an iterative database construction as follows:

[4] As was the case with one-shot sanitizers, since S is an arbitrary data structure, we must specify a way to compute the answer to a query $q \in \mathcal{Q}$ from the approximation S. In all our applications, this procedure will be clear from context and we will abuse notation by writing $q(S)$ to indicate that this procedure should be applied to S.

Definition 2.13 (Iterative Database Construction)**.** Let $\mathbf{U} : \mathcal{S} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{S}$ be an update function. We say $\mathbf{U}$ is an *iterative database construction with mistake bound B for query family $\mathcal{Q}$* (where B may depend on the parameters of the IDC) if for every database $D \in (\{0,1\}^d)^n$, every $(\mathbf{U}, D, \mathcal{Q}, \alpha, C)$-database update sequence satisfies $C \leq B$.

Note that the definition of $\mathbf{U}$ is such that if $\mathbf{U}$ is an iterative database construction with mistake bound B, then given any maximal $(\mathbf{U}, D, \mathcal{Q}, \alpha, C)$-database update sequence, the final approximation $S^{(C)}$ must satisfy

$$\forall q \in \mathcal{Q},\ \left|q(D) - q(S^{(C)})\right| \leq \alpha.$$

Otherwise, there would exist another query satisfying property 2 of Definition 2.12, contradicting maximality.

We can obtain the following theorem by combining any IDC with the sparse vector technique. For a set of queries Q from a family of queries $\mathcal{Q}$, and let $\mathcal{M}_{\mathrm{IDC}}(D, Q)$ be the following: processes queries $q \in Q$ in an arbitrary order, for each query q_j define a function $f_j(D) = |q_j(D) - q_j(S^{(t)})|$, where $S^{(t)}$ will be the current approximation used by the IDC when the j-th query is considered. Run the sparse vector algorithm on D and the set of queries $F = \{f_j\}$, with accuracy parameters 2α and β, and $\ell = B$. After each round in which $f_j(D)$ has a large answer, use q_j as the distinguishing query for $\mathbf{U}$ to obtain a new approximation $S^{(t+1)}$. Since the IDC is publicly known, the values $S^{(t)}$ can be computed by the analyst and thus the value $|q_j(D) - q_j(S^{(t)})| + z_j$ is sufficient for the analyst to recover an accurate answer to $q_j(D)$. We remark that typically the step of deriving an approximation to $q_j(D)$ would be incorporated into the specification of the algorithm.

We can summarize the privacy and accuracy of this approach with the following theorems:

Theorem 2.14. *The algorithm $\mathcal{M}_{\mathrm{IDC}}(D, Q)$ defined above is (ε, δ)-differentially private.*

Theorem 2.15. *Let $\alpha > 0$ be a parameter and $\mathcal{Q}$ be a family of counting queries. If there is an iterative database construction, $\mathbf{U}$, with mistake bound B for $\mathcal{Q}$ then $\mathcal{M}_{\mathrm{IDC}}(D, Q)$ defined above is an (ε, δ)-differentially private sanitizer that is $(4\alpha, \beta)$-accurate for any set of queries Q from $\mathcal{Q}$ and runs in time $\mathrm{poly}(T_{\mathbf{U}}, d, n, |q_1| + \cdots + |q_k|)$ so long as*

$$n \geq \frac{16\sqrt{B}\log(|Q|/\beta)\log(4/\delta)}{\alpha\varepsilon}.$$

Here $T_{\mathbf{U}}$ is the running time of $\mathbf{U}$.

The privacy of the IDC construction follows from privacy of the sparse vector technique (Lemma 2.11). Accuracy will follow from the mistake bound of the IDC, which will ensure that the number of functions f_j such that $f_j(D)$ is large will be at most B. The requirement that n be sufficiently large ensures that the noise added by the sparse vector algorithm is sufficiently small. Specifically, if the noise is sufficiently small, then the set of queries q_j such that $f_j(D)$ is large will constitute a database update sequence and the mistake bound of the IDC will apply.

In Chapter 6 we will construct a new IDC for the family of k-way marginal queries, and thus obtain a new sanitizer for k-way marginal queries via Theorem 2.15. In Chapter 7 we will design a new sanitizer achieving analyst differential privacy and we will use the fact that it can be cast in the IDC framework to argue that it satisfies differential privacy for the data subjects.

2.4.4 The Private Multiplicative Weights Algorithm

The Laplace mechanism is quite powerful, and is tough to match for simplicity and efficiency. However, it is quite limited in that it can answer at most quadratically many counting queries while maintaining privacy and non-trivial accuracy. One of the major achievements of differential privacy has been the discovery of (inefficient) sanitizers that accurately answer exponentially many counting queries. The first such algorithm was due to Blum, Ligett, and Roth [16]. The strongest quantitative guarantees in terms of running time and accuracy are given by variants of the private multiplicative weights algorithm of Hardt and Rothblum [44], and so we will use this algorithm as a reference point. The private multiplicative weights algorithm will be used as a tool for obtaining algorithms satisfying analyst differential privacy in Chatper 7, and a variant of the algorithm will be used in Chapter 6 to give faster sanitizers for answering marginal queries. Private multiplicative weights also serves as a foil for the hardness results of Chatpers 3 and 4.

The following lemma summarizes the properties of the private multiplicative weights algorithm, using the improved analysis from Gupta et al. [42].

Lemma 2.16. *Let $D \in (\{0,1\}^d)^n$ be a database and $Q = \{q_1, \ldots, q_k\}$ be a set of $(1/n)$-sensitive counting queries, $q_j \colon (\{0,1\}^d)^n \to [0,1]$. Let $\mathcal{M}_{\mathrm{MW}}(D, Q)$ be the private multiplicative weights algorithm. Then the following all hold:*

1) $\mathcal{M}_{\mathrm{MW}}$ is (ε, δ)-differentially private.

2) For any $\beta > 0$, $\mathcal{M}_{\mathrm{MW}}$ is (α, β)-accurate for

$$\alpha = O\left(d^{1/4} \frac{\sqrt{\log(k/\beta)\log(1/\delta)}}{\sqrt{\varepsilon n}}\right),$$

and

3) $\mathcal{M}_{\mathrm{MW}}$ runs in time $\mathrm{poly}(2^d, n, k, |q_1| + \cdots + |q_k|)$.

In our terminology, we can summarize the properties of the private multiplicative weights algorithm with the following corollary:

Corollary 2.17. *The private multiplicative weights mechanism $\mathcal{M}_{\mathrm{MW}}$ is 1) differentially private, 2) is accurate for $2^{\widetilde{\Omega}(n)/\sqrt{d}}$ queries from $\mathcal{Q}_{\mathsf{all}}^{(d)}$, and, 3) on input a set of queries $q_1, \ldots, q_k \in \mathcal{Q}_{\mathsf{all}}^{(d)}$, runs in time $\mathrm{poly}(2^d, n, |q_1| + \cdots + |q_k|)$.*

Chapter 3

Answering $n^{2+o(1)}$ Arbitrary Counting Queries is Hard

In this chapter we give new hardness results for differentially private sanitizers that answer a large number of queries. Recall from Chapter 2 that the best *efficient* sanitizer, the Laplace mechanism, is capable of answering nearly n^2 arbitrary counting queries with non-trivial accuracy. On the other hand, inefficient algorithms such as private multiplicative weights [44] can accurately answer nearly 2^n queries, and it is information-theoretically impossible to answer more than 2^n queries [26]. Thus, for efficient sanitizers there is a huge gap in our knowledge between n^2 and 2^n queries, and (under cryptographic assumptions) this gap is almost completely closed by the results of this chapter.

3.1 Results and Techniques

We prove the following new hardness results for answering counting queries while satisfying differential privacy.

Theorem 3.1. *Assuming the existence of one-way functions, there is no efficient, differentially private sanitizer that is accurate for $n^{2+o(1)}$ queries from $\mathcal{Q}_{\mathsf{all}}$.*

That is, there is no algorithm that, on input a database $D \in (\{0,1\}^d)^n$ and $n^{2+o(1)}$ efficiently ($\mathrm{poly}(d,n)$-time) computable counting queries, runs in time $\mathrm{poly}(d,n)$ and returns an approximate answer to each query to within $\pm 1/3$, while satisfying differential privacy.

Recall that we use $\mathcal{Q}_{\mathsf{all}}$ to denote the set of all counting queries. The choice of $1/3$ in the conclusion is arbitrary, and can be replaced with any constant less than $1/2$.

In particular, Theorem 3.1 applies to *online sanitizers*, which are sanitizers that receive (possibly adaptively chosen) queries one at a time. Many positive results achieve this stronger notion of sanitization. Although we have not presented it as such, the Laplace mechanism is an efficient online sanitizer that answers $\widetilde{\Omega}(n^2)$ queries and the private multiplicative weights algorithm is an online sanitizer that can answer nearly 2^n queries in time $\mathrm{poly}(2^d, n)$ per query [78, 44, 42].

We also show that the same theorem holds even for queries that are computable by unbounded-fan-in circuits of depth 6 over the basis $\{\wedge, \vee, \neg\}$ (a subset of the well-studied class AC^0), albeit under a strong (but plausible) cryptographic assumption.

Theorem 3.2. *Under the assumptions described in Section 3.4.6, there is no efficient, differentially private sanitizer that is accurate for $n^{2+o(1)}$ queries from $\mathcal{Q}_{\mathsf{depth}-6}$.*

That is, there is no algorithm that, on input a database $D \in (\{0,1\}^d)^n$ and $n^{2+o(1)}$ efficiently ($\mathrm{poly}(d, n)$-time) computable depth-6 queries (circuits), runs in time $\mathrm{poly}(d, n)$ and returns an approximate answer to each query to within $\pm 1/3$, while satisfying differential privacy.

Recall that, for a constant $h \in \mathbb{N}$, $\mathcal{Q}_{\mathsf{depth}-h}$ to denotes the set of all counting queries specified by circuits of depth h.

In Chapters 5 and 6, we will see that these hardness results can be circumvented by only requiring the sanitizer to answer marginal queries. In Chapter 5 we present an efficient (one-shot) sanitizer that answers $n^{\Omega(\sqrt{k})} \gg n^2$ many k-way marginal queries. And in Chapter 6 give a sanitizer that answers up to an exponential number ($2^{\widetilde{\Omega}(n)/d^{.51}}$) of arbitrary marginal queries accurately with running time $\mathrm{poly}(2^{o(d)}, n, k)$.

We now describe our techniques.

3.1.1 The Connection with Traitor-Tracing

We prove our results by building on the connection between differentially private sanitizers for counting queries and *traitor-tracing schemes* discovered by Dwork et al. [32]. Traitor-tracing schemes were introduced by Chor, Fiat, and Naor [23] for the purpose of identifying pirates who violate copyright restrictions. Roughly speaking, a (fully collusion-resilient) traitor-tracing scheme allows a sender to generate keys for n users so that 1) the sender can broadcast encrypted messages

that can be decrypted by any user, and 2) any *efficient pirate decoder* capable of decrypting messages can be *traced* to at least one of the users who contributed a key to it, even if an arbitrary coalition of the users combined their keys in an arbitrary efficient manner to construct the decoder.

Dwork et al. show that the existence of traitor-tracing schemes implies hardness results for one-shot sanitizers. Very informally, they argue as follows: Suppose a coalition of users takes their keys and builds a database $D \in (\{0,1\}^d)^n$ where each record contains one of their user keys. The family $\mathcal{Q}$ will contain a query q_c for each possible ciphertext c. The query q_c asks "What fraction of the records (user keys) in D would decrypt the ciphertext c to the message 1?" Every user can decrypt, so if the sender encrypts a message $m \in \{0,1\}$ as a ciphertext c, then every user will decrypt c to m. Thus the answer to the counting query, q_c, will be m.

Suppose there were an efficient one-shot sanitizer for $\mathcal{Q}$. Then the coalition could use it to efficiently produce a summary of the database D that enables one to efficiently compute an approximate answer to every query q_c, which would also allow one to efficiently decrypt the ciphertext. Such a summary can be viewed as an efficient pirate decoder, and thus the tracing algorithm can use the summary to trace one of the users in the coalition. However, if there is a way to identify one of the users in the database from the summary, then the summary is not differentially private.

In order to instantiate their result, they need a traitor-tracing scheme. Since $\mathcal{Q}$ contains a query for every ciphertext, the parameter to optimize is the length of the ciphertexts. Using the fully collusion-resilient traitor-tracing scheme of Boneh, Sahai, and Waters [17], which has ciphertexts of length $\widetilde{O}(\sqrt{n})$, they obtain a family of queries of size $2^{\widetilde{O}(\sqrt{n})}$ for which there is no efficient one-shot sanitizer. Dwork et al. also discovered a partial converse—proving hardness of one-shot sanitization for a smaller family of queries requires constructing traitor-tracing schemes with shorter ciphertexts, which is a seemingly difficult open problem.

3.1.2 Our Approach

In our setting of sanitization (rather than one-shot sanitization, as studied by Dwork et al. [32]), we don't expect to answer every query in $\mathcal{Q}$, only a much smaller set of queries requested by the analyst. At first glance, this should make answering the queries much easier, and thus make it more difficult to demonstrate hardness. However, the attacker does have the power to choose the queries that he wants answered, and can choose queries that are most difficult to sanitize. Our first observation is that in the traitor-tracing scenario, the tracing algorithms only query the pirate

decoder on a polynomial number of ciphertexts, which are randomly chosen and depend on the particular keys that were instantiated for the scheme. For many schemes, even $\widetilde{O}(n^2)$ queries is sufficient. Thus it would seem that the tracing algorithm could simply decide which queries it will make, give those queries as input to the sanitizer, and then use the answers to those queries to identify a user and violate differential privacy.

However, this intuition ignores an important issue. Many traitor-tracing schemes (including [17]) can only trace *stateless* pirate decoders, which essentially commit to a response to each possible query (or a distribution over responses) once and for all. For one-shot sanitizers, the private summary is necessarily stateless, and thus the result of Dwork et al. can be instantiated with any scheme that allows tracing of stateless pirate decoders. However, an arbitrary sanitizer might give answers that depend on the sequence of queries. Thus, in order to prove our results, we will need a traitor-tracing scheme that can trace *stateful* pirate decoders.

The problem of tracing stateful pirates is quite natural even without the implications for private data analysis. Indeed, this problem has been studied in the literature, originally by Kiayias and Yung [54]. They considered pirates that can *abort* and *record history*. However, their solution, and all others known, does not apply to our specific setting due to a certain "watermarking assumption" that doesn't apply when proving hardness-of-sanitization (see discussion below). To address this problem, we also refine the basic connection between traitor-tracing schemes and differential privacy by showing that, in many respects, fairly weak traitor-tracing schemes suffice to establish the hardness of preserving privacy. In particular, although the pirate decoder obtained from a sanitizer may be stateful and record history, the accuracy requirement of the sanitizer means that the corresponding pirate decoder cannot abort, which will make it easier to construct a traitor-tracing scheme for these kinds of pirates. Indeed, we construct such a scheme to establish Theorem 6.1.

The scheme also has weakened requirements in other respects, having nothing to do with the statefulness of the pirate or the tracing algorithm. These weakened requirements allow us to reduce the complexity of the decryption, which means that the queries used by the attacker do not need to be arbitrary polynomial-size circuits, but instead can be circuits of constant depth, which allows us to establish Theorem 6.3. Another technical issue arises in that all k queries must be given to the sanitizer at once, whereas tracing algorithms typically are allowed to query the pirate interactively. However, we are able to show that the scheme we construct can be traced using one round of queries. See Sections 3.2.1 and 3.3 for a precise statement of the kind of traitor-tracing scheme

that suffices and Section 3.4 for our construction.

Our construction is based on a well-known fully collusion resilient traitor-tracing scheme [23], but with a modified tracing algorithm. The tracing algorithm uses *fingerprinting codes* [18, 86], which have been employed before in the context of traitor-tracing and content distribution, but our tracing algorithm is different from all those we are aware of. The resulting scheme allows for tracing with a minimal number of non-adaptively chosen queries, achieves tracing without context-specific watermarking assumptions, simplifies the decryption circuit (at the expense of weakening the security parameters and functionality). The restriction to non-aborting pirates may not be so natural in the setting of content distribution, which may explain why the scheme was not previously known.

3.1.3 Related Work

In addition to the hardness results for one-shot sanitizations [32], which apply to arbitrary one-shot sanitizers, there are several hardness-of-sanitization results for restricted classes of sanitizers. As we will show in Chapter 4 (building on earlier work of Dwork et al. [32], it is hard (assuming the existence of one-way functions) to generate a private synthetic database that is accurate for essentially any non-trivial family of queries, even 2-literal conjunctions. Recall from Chapter 2 that a synthetic database is, roughly, a new database (of the same dimensions) that approximately preserves the answer to some set of queries.

Gupta et al. [41] considered algorithms that can only access the database by making a polynomial number of "statistical queries" (essentially counting queries). They showed that such algorithms cannot be a one-shot sanitizer (even ignoring privacy constraints) that approximately answers certain simple families of counting queries with high accuracy.

Finally, Dwork, Naor, and Vadhan [33] gave information-theoretic lower bounds for *stateless sanitizers*, which take k queries as input, but whose answers to each query do not depend on the other $k-1$ input queries. They showed that (even computationally unbounded) stateless sanitizers can answer at most $\widetilde{O}(n^2)$ queries with non-trivial accuracy, while satisfying differential privacy. The Laplace Mechanism is a stateless sanitizer that answers $\widetilde{\Omega}(n^2)$ queries, and thus their result is tight in this respect. Although their result is information theoretic, and considers a highly restricted type of sanitizer, their techniques are related to ours.

3.2 Traitor-Tracing Schemes

In this section we give define a traitor-tracing scheme. Throughout, we will use A_{TT} to denote algorithms associated with traitor-tracing schemes.

3.2.1 Traitor-Tracing Schemes

We now give a definition of a traitor-tracing scheme, heavily tailored to the task of proving hardness results for generic sanitizers. We will sacrifice some consistency with the standard definitions. See below for further discussion of the ways in which our definition departs from the standard definition of traitor-tracing. In some cases, the non-standard aspects of the definition will be necessary to establish our results, and in others it will be for convenience. Despite these differences, we will henceforth refer to schemes satisfying our definition simply as *traitor-tracing schemes*.

Intuitively, a traitor-tracing scheme is a form of broadcast encryption, in which a sender can broadcast an encrypted message that can be decrypted by each of a large set of users. The standard notion of security for such a scheme would require that an adversary that doesn't have any of the keys cannot decrypt the message. A traitor-tracing scheme has the additional property that given any decoder capable of decrypting the message (which must in a very loose sense "know" at least one of the keys), there is a procedure for determining which user's key is being used. Moreover, we want the scheme to be "collusion resilient," in that even if a coalition of users gets together and combines their keys in some way to produce a decoder, there is still a procedure that identifies at least one member of the coalition.

We now formally describe the syntax of a traitor-tracing scheme. For functions $n, k_{\mathsf{TT}} \colon \mathbb{N} \to \mathbb{N}$, an (n, k_{TT})-traitor-tracing scheme is a tuple of four algorithms $(\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$. The parameter n specifies the number of users of the scheme and the parameter k_{TT} will specify the number of queries that the tracing algorithm makes to the pirate decoder. We allow all the algorithms to be randomized except for $\mathsf{Dec}_{\mathsf{TT}}$.[5]

- The algorithm $\mathsf{Gen}_{\mathsf{TT}}$ takes a security parameter, κ, and returns a sequence of $n = n(\kappa)$ user keys $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)}) \in \{0,1\}^{\kappa}$. Formally, $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)}) \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^{\kappa})$.

[5]It would not substantially affect our results if $\mathsf{Dec}_{\mathsf{TT}}$ were randomized, but we will assume that $\mathsf{Dec}_{\mathsf{TT}}$ is deterministic for ease of presentation.

- The algorithm $\mathsf{Enc}_{\mathsf{TT}}$ takes a sequence of n user keys $\vec{sk}$ and a message bit $b \in \{0,1\}$, and generates a ciphertext $c \in \mathcal{C} = \mathcal{C}^{(\kappa)}$. Formally, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}_{\mathsf{TT}}(\vec{sk}, b)$.

- The algorithm $\mathsf{Dec}_{\mathsf{TT}}$ takes any single user key sk and a ciphertext $c \in \mathcal{C}$, runs in time $\mathrm{poly}(\kappa, n(\kappa))$ and deterministically returns a message bit $\widehat{b} \in \{0,1\}$. Formally we write $\widehat{b} = \mathsf{Dec}_{\mathsf{TT}}(sk, c)$.

- The algorithm $\mathsf{Trace}_{\mathsf{TT}}$ takes as input a set of user keys $\vec{sk} \in (\{0,1\}^{\kappa})^{n(\kappa)}$ and an oracle $\mathcal{P}\colon (\mathcal{C}^{(\kappa)})^{k_{\mathsf{TT}}(\kappa)} \to \{0,1\}^{k_{\mathsf{TT}}(\kappa)}$, makes one k_{TT}-tuple of queries, $(c_1, \ldots, c_{k_{\mathsf{TT}}}) \in \mathcal{C}^{(\kappa)}$ to its oracle ($k_{\mathsf{TT}} = k_{\mathsf{TT}}(\kappa)$), and returns the name of a user $i \in [n(\kappa)]$. Formally, $i \leftarrow_{\mathrm{R}} \mathsf{Trace}^{\mathcal{P}}_{\mathsf{TT}}(\vec{sk})$.

Intuitively, think of the oracle $\mathcal{P}$ as being given some subset of keys $\vec{sk}_S = (sk^{(i)})_{i \in S}$ for a non-empty set $S \subseteq [n]$, and $\mathsf{Trace}_{\mathsf{TT}}$ is attempting to identify a user $i \in S$. Clearly, if $\mathcal{P}$ ignores its input and always returns 0, $\mathsf{Trace}_{\mathsf{TT}}$ cannot have any hope of success, so we must assume that $\mathcal{P}$ is capable of decrypting ciphertexts.

Definition 3.3 (Available Pirate Decoder)**.** Let $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ be an (n, k_{TT})-traitor-tracing scheme. Let $\mathcal{P}$ be a (possibly randomized) algorithm. We say that $\mathcal{P}$ is a k_{TT}*-available pirate decoder* if for every $\kappa \in \mathbb{N}$, every set of user keys $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)}) \in \{0,1\}^{\kappa}$, every $S \subseteq [n]$ such that $|S| \geq n - 1$, and every $c_1, \ldots, c_{k_{\mathsf{TT}}} \in \mathcal{C}^{(\kappa)}$,

$$\Pr\left[\begin{array}{c}(\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}}) \leftarrow_{\mathrm{R}} \mathcal{P}(\vec{sk}_S, c_1, \ldots, c_{k_{\mathsf{TT}}}) \\ \exists j \in [k_{\mathsf{TT}}], b \in \{0,1\}\left((\forall i \in S, \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = b) \wedge \left(\widehat{b}_j \neq b\right)\right)\end{array}\right] \leq o\left(\frac{1}{n(\kappa)^2}\right).$$

In other words, if every user key $sk^{(i)}$ (for $i \in S$) decrypts c to 1 (resp. 0), then $\mathcal{P}(\vec{sk}_S, \cdot)$ decrypts c to 1 (resp. 0), with high probability.

We can now define a secure, (n, k_{TT})-traitor-tracing scheme:

Definition 3.4 (Traitor-Tracing Scheme)**.** Let the scheme $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ be an (n, k_{TT})-traitor-tracing scheme. Let $k_{\mathsf{TT}}\colon \mathbb{N} \to \mathbb{N}$ be a function. We say that Π_{TT} is a *secure* (n, k_{TT})*-traitor-tracing scheme* if for every $S \subseteq [n(\kappa)]$ such that $|S| \geq n(\kappa) - 1$, for every (possibly randomized) algorithm $\mathcal{P}$ that 1) runs in time $\mathrm{poly}(\kappa, n(\kappa), k_{\mathsf{TT}}(\kappa))$ and 2) is a k_{TT}-available pirate decoder, we have

$$\Pr_{\substack{\vec{sk} \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^{\kappa}) \\ \mathcal{P}\text{'s, } \mathsf{Trace}_{\mathsf{TT}}\text{'s coins}}}\left[\mathsf{Trace}^{\mathcal{P}(\vec{sk}_S, \cdot)}_{\mathsf{TT}}(\vec{sk}) \notin S\right] = o\left(\frac{1}{n(\kappa)}\right)$$

Remarks About Our Definition of Traitor-Tracing The traitor-tracing schemes we consider are somewhat different than those previously studied in the literature.

- We do not require the encryption or tracing algorithms to use public keys. In the typical application of traitor-tracing schemes to content distribution, these would be desirable features, however they are not necessary for proving hardness of sanitization.

- We only require that the tracing algorithm succeeds with probability $1 - o(1/n)$. Typically one would require that the tracing algorithm succeeds with probability $1 - n^{-\omega(1)}$.

- We do not give the pirate decoder access to an encryption oracle. In other words, we do not require CPA security. Most traitor-tracing schemes in the literature are public-key, making this distinction irrelevant. Here, we only need an encryption scheme that is secure for an *a priori* bounded number of messages.

- We allow the pirate decoder to be *stateful*, but in an unusual way. In many other models, the pirate is allowed to abort, and answer $\perp$ if it detects that it is being traced. However, in our model we require (roughly) that if any of the queries are ciphertexts generated by $\mathsf{Enc}(\vec{sk}, b)$, then the pirate decoder answers b to those queries, regardless of the other queries issued, which in a sense precludes aborting. However, we do allow our pirate to correlate its answers to different queries, subject to this accuracy constraint. We also allow the pirate to see all the queries made by the tracer at once, which is more power than is typically given to the pirate.

Roughly, the first three modifications will allow us to find a candidate scheme with very simple decryption and the fourth modification will allow us to trace stateful pirates even in the setting of bit-encryption.

3.2.2 Decryption Function Families

For Theorem 6.3, we are interested in traitor-tracing schemes where $\mathsf{Dec}_{\mathsf{TT}}$ is a "simple" function of the user key (for every ciphertext $c \in \mathcal{C}$).

Definition 3.5 (Decryption Function Family)**.** Let $(\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}})$ be any traitor-tracing scheme where $\mathsf{Gen}_{\mathsf{TT}}$ produces keys in $\{0,1\}^\kappa$ and $\mathsf{Enc}_{\mathsf{TT}}$ produce ciphertexts in $\mathcal{C} = \mathcal{C}^{(\kappa)}$. For

every $c \in \mathcal{C}$, we define the *c-decryption function* $q_c \colon \{0,1\}^\kappa \to \{0,1\}$ to be $q_c(sk) = \mathsf{Dec}_{\mathsf{TT}}(sk, c)$. We define the *decryption function family* $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}} = \{q_c\}_{c \in \mathcal{C}^{(\kappa)}}$.

In what follows, we will say that Π_{TT} is an traitor-tracing scheme with decryption function family $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$.

3.3 Attacking Efficient Sanitizers

In this section we will prove our main result, showing that the existence of traitor-tracing schemes (as in Definition 3.4) implies that efficient sanitizers cannot answer too many counting queries while satisfying differential privacy.

Theorem 3.6 (Attacking Efficient Sanitizers)**.** *Assume that there exists an $(n(\kappa), k_{\mathsf{TT}}(\kappa))$-secure traitor-tracing scheme $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ with decryption function family $\mathcal{Q}^{(\kappa)} = \mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$. Then there does not exist any sanitizer $\mathcal{M} \colon (\{0,1\}^d)^n \times (\mathcal{Q}^{(d)})^{k_{\mathsf{TT}}(d)} \to \mathbb{R}^{k_{\mathsf{TT}}(d)}$ that is simultaneously 1) differentially private, 2) efficient, and 3) accurate for $k_{\mathsf{TT}}(d)$ queries from $\mathcal{Q} = \cup_{d \in \mathbb{N}} \mathcal{Q}^{(d)}$.*

In the typical setting of parameters, $n(\kappa) = \mathrm{poly}(\kappa)$, $k_{\mathsf{TT}}(\kappa) = \widetilde{\Theta}(n^2)$, and decryption can be implemented by circuits of size $\mathrm{poly}(n) = \mathrm{poly}(\kappa)$. Then Theroem 3.6 will state that there is no sanitizer $\mathcal{M}$ that takes a database $D \in (\{0,1\}^d)^{\mathrm{poly}(d)}$, runs in $\mathrm{poly}(d)$ time, and accurately answers $\widetilde{\Theta}(n^2)$ queries implemented by circuits of size $\mathrm{poly}(d)$, while satisfying differential privacy.

The main difference between Theorem 3.6 and the result of Dwork et al. [32] is that we only assume the existence of a sanitizer for $k_{\mathsf{TT}}(d)$ queries from $\mathcal{Q}^{(d)} = \mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}$, whereas Dwork et al. assume the existence of a one-shot sanitizer that answers every query in $\mathcal{Q}^{(d)}$. To offset the weaker assumption on the sanitizer, we assume that the traitor-tracing scheme is secure against certain stateful pirate decoders (as in Definition 3.3) whereas Dwork et al. only need to trace stateless pirates. Theorem 3.6 also explicitly allows the traitor-tracing scheme to have the relaxed functionality and security properties discussed at the end of Section 3.2, although it is implicit in Dwork et al. that the relaxed properties are sufficient to prove hardness results.

We now sketch the proof: Every function $q_c \in \mathcal{Q}^{(d)}$ is viewed as a query $q_c(x)$ on a database row $x \in \{0,1\}^d$. Assume there is an efficient sanitizer is that is accurate for $k_{\mathsf{TT}}(d)$ queries from $\mathcal{Q}^{(d)}$. The fact that $\mathcal{M}$ is accurate for these queries will imply that (after small modifications)

$\mathcal{M}$ is a k_{TT}-available pirate decoder (Definition 3.3). Here is where we differ from Dwork et al., who assume that $\mathcal{M}$ accurately answers *all* queries in $\mathcal{Q}^{(d)}$, in which case $\mathcal{M}$ can be viewed as a stateless pirate decoder (but must solve a harder sanitization problem).

We complete the proof as in Dwork et al. Consider two experiments: In the first, we construct an n-row database D by running $\mathsf{Gen}_{\mathsf{TT}}(1^d)$ to obtain n user keys, and putting one in each row of D. Then we run $\mathsf{Trace}_{\mathsf{TT}}$ on $\mathcal{M}(D, \cdot)$ and obtain a user i. Since $\mathcal{M}$ is useful, and Π_{TT} is secure, we will have that $i \in [n]$ with probability close to 1, and thus there is an $i^* \in [n]$ such that $i = i^*$ with probability $\gtrsim 1/n$.

In the second experiment, we construct a database D' exactly as in the first, however we exclude the key $sk^{(i^*)}$. Since D and D' differ in only one row, differential privacy requires that $\mathsf{Trace}_{\mathsf{TT}}$, run with oracle $\mathcal{M}(D', \cdot)$, still outputs i^* with probability $\Omega(1/n)$. However, in this experiment, i^*, $sk^{(i^*)}$ is no longer given to the pirate decoder, and thus security of Π_{TT} says that $\mathsf{Trace}_{\mathsf{TT}}$, run with this oracle, must output i^* with probability $o(1/n)$. Thus, we will obtain a contradiction.

Proof. Let $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ be the assumed traitor-tracing scheme, and assume there exists an efficient, differentially private, sanitizer $\mathcal{M}$ that is accurate for $k_{\mathsf{TT}}(d)$ queries from $\mathcal{Q}^{(d)}$. We define the pirate decoder $\mathcal{P}_{\mathcal{M}}$ as follows:

Input: A set of user keys $(\vec{sk}_S) \in \{0,1\}^d$ and a sequence of k_{TT} ciphertexts $c_1, \ldots, c_{k_{\mathsf{TT}}}$.
Construct circuits specifying the queries $q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}} \in \mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}},d}$.
Construct a database $D = (sk^{(i)})_{i \in S} \in (\{0,1\}^d)^{|S|}$.
Let $a_1, \ldots, a_{k_{\mathsf{TT}}} \leftarrow_{\mathrm{R}} \mathcal{M}(D, q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}})$.
Round the answers $a_1, \ldots, a_{k_{\mathsf{TT}}} \in [0,1]$ to obtain $\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}} \in \{0,1\}$ (i.e. $\widehat{b}_j = \lceil a_j \rfloor$)
Output: $\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}}$.

Figure 3: The pirate decoder $\mathcal{P}_{\mathcal{M}}$

Since $\mathcal{M}$ is efficient, its running time is $\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d), |q_{c_1}| + \ldots + |q_{c_{k_{\mathsf{TT}}}}|)$, which is $\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d))$. Recall that the size of the circuits (queries) $q_c \in \mathcal{Q}^{(d)}_{\mathsf{Dec}_{\mathsf{TT}}}$ is $\mathrm{poly}(d, n)$. In this case $\mathcal{P}_{\mathcal{M}}$ runs in time $\mathrm{poly}(d, n(d), k_{\mathsf{TT}}(d))$ as well, since constructing the queries can be done in time polynomial in their size, and the final rounding step can be done in time $\mathrm{poly}(k_{\mathsf{TT}}(d))$.

Next, we claim that if $\mathcal{M}$ is accurate for $\mathcal{Q}^{(d)}$, then $\mathcal{P}_{\mathcal{M}}$ is a useful pirate decoder.

Claim 3.7. *If $\mathcal{M}$ is accurate for k_{TT} queries from $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}}}$, then $\mathcal{P}_{\mathcal{M}}$ is a k_{TT}-useful pirate decoder.*

Proof of Claim 3.7. Let $\vec{sk} \in \{0,1\}^d$ be a set of user keys for Π_{TT} and let $S \subseteq [n]$ be a subset of the users such that $|S| \geq n-1$. Suppose $c \in \mathcal{C}^{(d)}$ and $b \in \{0,1\}$ are such that for every $i \in S$, $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c) = b$. Then we have that, for D as in $\mathcal{P}_{\mathcal{M}}$,

$$q_c(D) = \frac{1}{|S|}\sum_{i\in S} q_c(sk^{(i)}) = \frac{1}{|S|}\sum_{i\in S} \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c) = b$$

Let $c_1, \ldots, c_{k_{\mathsf{TT}}}$ be a set of ciphertexts, $q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}}$ and $a_1, \ldots, a_{k_{\mathsf{TT}}}$ be as in $\mathcal{P}_{\mathcal{M}}$. The accuracy of $\mathcal{M}$ (with constant error $\alpha < 1/2$) guarantees that

$$\Pr\left[\exists j \in [k_{\mathsf{TT}}], \left|a_j - f_{c_j}(D)\right| \geq 1/2\right] = o(1/|S|^2)$$

Since $|S| \geq n-1$, $o(1/|S|^2) = o(1/n^2)$. Assuming $a_1, \ldots, a_{k_{\mathsf{TT}}}$ is accurate up to error $\alpha < 1/2$ for $q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}}$, a_j will be rounded to exactly q_{c_j} whenever $q_{c_j}(D) \in \{0,1\}$. That is,

$$\Pr\left[\begin{array}{c}\exists j \in [k_{\mathsf{TT}}], b \in \{0,1\} \\ (\forall i \in S, \mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = b) \wedge \left(\widehat{b}_j \neq b\right)\end{array}\right] = o\left(\frac{1}{n(\kappa)^2}\right)$$

Thus, $\mathcal{P}_{\mathcal{M}}$ is k_{TT}-useful. This completes the proof of the claim. □

Since $\mathcal{P}_{\mathcal{M}}$ is a k_{TT}-useful pirate decoder, and Π_{TT} is a (n, k_{TT})-secure traitor-tracing scheme, running $\mathsf{Trace}_{\mathsf{TT}}$ on $\mathcal{P}_{\mathcal{M}}$ will always return some user $i \in [n]$. Thus there must be some user i^* that $\mathsf{Trace}_{\mathsf{TT}}$ returns with probability $\gtrsim 1/n$. Specifically, for every $\kappa \in \mathbb{N}$, there exists $i^*(\kappa) \in [n(\kappa)]$ such that,

$$\Pr_{\substack{\vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}(1^\kappa) \\ \mathcal{P}_{\mathcal{M}}\text{'s, } \mathsf{Trace}_{\mathsf{TT}}\text{'s coins}}}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}_{\mathcal{M}}(\vec{sk},\cdot)}(\vec{sk}) = i^*(\kappa)\right] \geq \frac{1}{n(\kappa)} - o\left(\frac{1}{n(\kappa)}\right). \tag{3.1}$$

Let $S(\kappa) = [n(\kappa)] \setminus \{i^*(\kappa)\}$ Now we claim that if $\mathcal{M}$ is differentially private, then $\mathsf{Trace}_{\mathsf{TT}}$ will output $i^*(\kappa)$ with significant probability, even $\mathcal{P}_{\mathcal{M}}$ is not given the key of user $i^*(\kappa)$.

Claim 3.8. *If $\mathcal{M}$ is differentially private (for $\varepsilon = O(1)$, $\delta = o(1/n)$), then*

$$\Pr_{\substack{\vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}(1^\kappa) \\ \mathcal{P}_{\mathcal{M}}\text{'s, } \mathsf{Trace}_{\mathsf{TT}}\text{'s coins}}}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}_{\mathcal{M}}(\vec{sk},\cdot)}(\vec{sk}) = i^*(\kappa)\right] \geq \Omega\left(\frac{1}{n(\kappa)}\right).$$

Proof of Claim 3.8. Fix any κ and let $k_{\mathsf{TT}} = k_{\mathsf{TT}}(\kappa)$ and $i^* = i^*(\kappa)$, $S = S(\kappa)$ as above. Let $D = \vec{sk}$ and $D_{-i^*} = \vec{sk}_S$. Take T to be the set of responses $\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}}$ such that $\mathsf{Trace}_{\mathsf{TT}}(\vec{sk})$,

after querying its oracle on ciphertexts $c_1, \ldots, c_{k_{\mathsf{TT}}}$ and receiving responses $\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}}$, outputs i^* (T depends on the coins of $\mathsf{Gen}_{\mathsf{TT}}$ and $\mathsf{Trace}_{\mathsf{TT}}$). By differential privacy, we have that

$$\Pr\left[\mathcal{M}(D, q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}}) \in T\right] \leq e^{O(1)} \cdot \Pr\left[\mathcal{M}(D_{-i^*}, q_{c_1}, \ldots, q_{c_{k_{\mathsf{TT}}}}) \in T\right] + o\left(\frac{1}{n}\right).$$

Note that the queries constructed by $\mathcal{P}_{\mathcal{M}}$ depends only on $c_1, \ldots, c_{k_{\mathsf{TT}}}$, not on $\vec{sk}_S$. Also note that the final rounding step does not depend on the input at all. Thus, for every $T \subseteq \{0,1\}^{k_{\mathsf{TT}}}$

$$\Pr\left[\mathcal{P}_{\mathcal{M}}(\vec{sk}, c_1, \ldots, c_{k_{\mathsf{TT}}}) \in T\right] \leq e^{O(1)} \cdot \Pr\left[\mathcal{P}_{\mathcal{M}}(\vec{sk}_S, c_1, \ldots, c_{k_{\mathsf{TT}}}) \in T\right] + o\left(\frac{1}{n}\right). \quad (3.2)$$

The claim follows by combining with (3.1). □

To complete the proof, notice that the probability in Claim 3.8 is exactly the probability that $\mathsf{Trace}_{\mathsf{TT}}$ outputs the user i^*, when given the oracle $\mathcal{P}_{\mathcal{M}}(\vec{sk}_S)$, for $S = [n] \setminus \{i^*\}$. However, the fact that $\mathcal{P}_{\mathcal{M}}$ is efficient, and Π_{TT} is a secure traitor-tracing scheme implies that this probability is $o(1/n)$. Thus we have obtained a contradiction. This completes the proof of the Theorem. □

3.4 Constructions of Traitor-Tracing Schemes

In this section we show how to construct traitor-tracing schemes that satisfy Definition 3.4, and thus can be used to instantiate Theorem 3.6. First we will informally describe a simple construction that requires the tracing algorithm to make a sub-optimal number of queries, but will hopefully give the reader more intuition about the construction and how it differs from previous constructions of traitor-tracing schemes. Then we will give precise definitions of the encryption schemes (Section 3.4.2) and fingerprinting codes (Section 3.4.3) required for our construction. Then we will present the final construction more formally (Section 3.4.4) and prove its security. Finally, we will use the weakened security requirements of the encryption scheme to show that our traitor-tracing scheme can be instantiated so that decryption is computable by constant-depth circuits (Section 3.4.6).

3.4.1 A Simple Construction

Our construction is a variant of the most basic tracing traitor-tracing scheme [23]. Start with an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Generate an independent key $sk^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}$ for each user

(we will ignore the security parameter in the informal description). To encrypt a bit $b \in \{0,1\}$, we encrypt it under each user's key independently and concatenate the ciphertexts. That is

$$\mathsf{Enc}_{\mathsf{TT}}(\vec{sk}, b) = (\mathsf{Enc}(sk^{(1)}, b), \ldots, \mathsf{Enc}(sk^{(n)}, b)).$$

Clearly each user can decrypt the ciphertext by applying Dec, as long as she knows which part of the ciphertext to decrypt.

Now we describe how an available pirate decoder for this scheme can be traced. As with all traitor-tracing schemes, we will form ciphertexts that different users would decrypt differently, assuming they decrypt as intended using the algorithm $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, \cdot)$. We can do so with the following algorithm:

$$\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i) = (\mathsf{Enc}(sk^{(1)}, 1), \ldots, \mathsf{Enc}(sk^{(i)}, 1), \mathsf{Enc}(sk^{(i+1)}, 0), \ldots, \mathsf{Enc}(sk^{(n)}, 0)$$

for $i = 0, 1, \ldots, n$. The algorithm forms a ciphertext that users $1, \ldots, i$ will decrypt to 0 and users $i+1, \ldots, n$ will decrypt to 1.

The tracing algorithm generates a random sequence $i_1, \ldots, i_{k_{\mathsf{TT}}} \in \{0, 1, \ldots, n\}$, for $k_{\mathsf{TT}} = (n+1)s$, such that each element of $\{0, 1, \ldots, n\}$ appears exactly s times, where s is a parameter to be chosen later. Then, for every j it generates a ciphertext $c_j \leftarrow_{\mathrm{R}} \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i_j)$. Next, it queries $\mathcal{P}_{\vec{sk}_S}(c_1, \ldots, c_{k_{\mathsf{TT}}})$. Given the output of the pirate, the tracing algorithm computes

$$P_i = \frac{1}{s} \sum_{j : i_j = i} \mathcal{P}(\vec{sk}, c_1, \ldots, c_{k_{\mathsf{TT}}})_j$$

for $i = 0, 1, \ldots, n$. Finally, the tracing algorithm outputs any i^* such that $P_{i^*} - P_{i^*-1} \geq 1/n$.

The tracing algorithm generates a random sequence of indices $i_1, \ldots, i_{k_{\mathsf{TT}}} \in \{0, 1, \ldots, n\}$, for $k_{\mathsf{TT}} = (n+1)s$, such that each element of $\{0, 1, \ldots, n\}$ appears exactly s times, where s is a parameter to be chosen later. Then, for every j it generates a ciphertext $c_j \leftarrow_{\mathrm{R}} \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i_j)$. Next, it queries $\mathcal{P}_{\vec{sk}_S}(c_1, \ldots, c_{k_{\mathsf{TT}}})$. Given the output of the pirate, the tracing algorithm computes $P_i = \frac{1}{s} \sum_{j : i_j = i} \mathcal{P}(\vec{sk}, c_1, \ldots, c_{k_{\mathsf{TT}}})_j$ for $i = 0, 1, \ldots, n$. Finally, the tracing algorithm outputs any i^* such that $P_{i^*} - P_{i^*-1} \geq 1/n$.

Now we explain why this algorithm successfully traces efficient available pirate decoders. Notice that if we choose c according to $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, 0)$, then every user decrypts c to 0, so $P_0 = 0$. Similarly, $P_n = 1$. Thus, there exists i^* such that $P_{i^*} - P_{i^*-1} \geq 1/n$. Next, we argue that i^* is in S except with small probability. Notice that $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*)$ and $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^* - 1)$ differ only in

the message encrypted under key $sk^{(i^*)}$, so if $i^* \notin S$, this key is unknown to the pirate decoder. The security of the encryption scheme (made precise in Definition 3.10) guarantees that if $sk^{(i^*)}$ is unknown to an efficient pirate, then we can replace k_{TT} uses of $\mathsf{Enc}(sk^{(i^*)}, 1)$ with $\mathsf{Enc}(sk^{(i^*)}, 0)$, and this change will only affect the success probability of the pirate by $o(1/n)$. But after we make this replacement, $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^*)$ and $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, i^* - 1)$ are (perfectly, information-theoretically) indistinguishable to the pirate. Since the sequence of indices $i_1, \ldots, i_{k_{\mathsf{TT}}}$ is random, the pirate has no information about which elements i_j are i^* and which are $i^* - 1$. Thus, if the pirate wants to make P_{i^*} larger than P_{i^*-1}, for some $i^* \notin S$, she can do no better than to "guess". If we take $s = \widetilde{O}(n^2)$, and apply a Chernoff bound, it turns out that for every $i \notin S$, $P_i - P_{i-1} = o(1/n)$. This conclusion also holds after we take into account the security loss of the encryption scheme, which is $o(1/n)$. Thus, the scheme we described is a secure traitor-tracing scheme in the sense of Definition 3.4.

In arguing that the scheme is secure, we used the fact that $P_0 = 0$ and $P_n = 1$ *no matter what other queries are made to the pirate*. In many applications, this assumption would not be reasonable. However, when the pirate is derived from an accurate sanitizer, this condition will be satisfied.

For this scheme, the tracer makes $(n+1)s = \widetilde{O}(n^3)$ queries. Before proceeding, we will explain how to reduce the number of queries from $\widetilde{O}(n^3)$ to $\widetilde{O}(n^2)$. The high-level argument that the scheme is secure used two facts:

1. By the availability of the pirate decoder, if every user in S would decrypt a ciphertext c to b, then the pirate decrypts c to b (in the above, $P_0 = 0, P_n = 1$).

2. Because of the encryption, a pirate decoder without user i's key "doesn't know" how user i would decrypt each ciphertext.

Systems leveraging these two properties to identify a colluding user are called *fingerprinting codes* [18], and have been studied extensively. In fact, the tracing algorithm we described is identical to the tracing algorithm we define in Section 3.4.4, but instantiated with the fingerprinting code of Boneh and Shaw [18], which has length $\widetilde{O}(n^3)$. Tardos [86] constructed shorter fingerprinting codes, with length $\widetilde{O}(n^2)$, which we use to reduce the number of queries to trace.

Next we define the precise security requirement we need out of the underlying encryption scheme, and then we will give a formal definition of fingerprinting codes.

3.4.2 Encryption Schemes

We will build our traitor-tracing scheme from a suitable encryption scheme. An encryption scheme is tuple of efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. All the algorithms may be randomized except for Dec. The scheme has the following syntactic properties:

- The algorithm Gen takes a security parameter κ, runs in time $\mathrm{poly}(\kappa)$, and returns a private key $sk \in \{0,1\}^\kappa$. Formally $sk \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)$.
- The algorithm Enc takes a private key and a message bit $b \in \{0,1\}$, runs in time $\mathrm{poly}(\kappa)$, and generates a ciphertext $c \in \mathcal{C} = \mathcal{C}^{(\kappa)}$. Formally, $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk, b)$.
- The algorithm Dec takes a private key $sk \in \{0,1\}^\kappa$ and a ciphertext $c \in \mathcal{C}^{(\kappa)}$, runs in time $\mathrm{poly}(\kappa)$, and returns a message bit $\widehat{b}$.

First we define (perfectly) correct decryption[6]

Definition 3.9 (Correctness)**.** An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *(perfectly) correct* if for every $b \in \{0,1\}$, and every $\kappa \in \mathbb{N}$,

$$\Pr_{sk \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)} [\mathsf{Dec}(sk, \mathsf{Enc}(sk, b)) = b] = 1.$$

We require that our schemes have the following k_{Enc}-message security property.

Definition 3.10 (Security of Encryption)**.** Let $\varepsilon_{\mathsf{Enc}} \colon \mathbb{N} \to [0,1]$ and $k_{\mathsf{Enc}} \colon \mathbb{N} \to \mathbb{N}, T_{\mathsf{Enc}} \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be functions. An encryption scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, T_{\mathsf{Enc}})$-*secure* if for every $T_{\mathsf{Enc}}(\kappa, k_{\mathsf{Enc}}(\kappa))$-time algorithm $\mathcal{A}_{\mathsf{Enc}}$ and every $b = (b_1, \ldots, b_{k_{\mathsf{Enc}}}), b' = (b'_1, \ldots, b'_{k_{\mathsf{Enc}}}) \in \{0,1\}$ (for $k_{\mathsf{Enc}} = k_{\mathsf{Enc}}(\kappa)$),

$$\left| \Pr_{sk \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)} [\mathcal{A}_{\mathsf{Enc}}(\mathsf{Enc}(sk, b_1), \ldots, \mathsf{Enc}(sk, b_{k_{\mathsf{Enc}}})) = 1] \right.$$
$$\left. - \Pr_{sk \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)} \left[\mathcal{A}_{\mathsf{Enc}}(\mathsf{Enc}(sk, b'_1), \ldots, \mathsf{Enc}(sk, b'_{k_{\mathsf{Enc}}})) = 1\right] \right| \leq \varepsilon_{\mathsf{Enc}}(\kappa).$$

Notice that we do not require Π_{Enc} to be secure against adversaries that are given $\mathsf{Enc}(sk, \cdot)$ as an oracle. That is, we do not require CPA security.

[6]It would not substantially affect our results if Dec were allowed to fail with negligible probability, however we will assume perfect correctness for ease of presentation.

Definition 3.11 (Encryption Scheme)**.** We say that a tuple of algorithms $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, T_{\mathsf{Enc}})$-encryption scheme if it satisfies correctness and $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, T_{\mathsf{Enc}})$-security.

3.4.3 Fingerprinting Codes

As we alluded to above, our tracing algorithm will use a *fingerprinting code*, introduced by Boneh and Shaw [18]. A fingerprinting code is a pair of efficient (possibly randomized) algorithms $(\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ with the following syntax.

- The algorithm $\mathsf{Gen}_{\mathsf{FP}}$ takes a number of users n as input and outputs a codebook of n codewords of length $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$, $W = (w^{(1)}, \ldots, w^{(n)}) \in \{0,1\}^{\ell_{\mathsf{FP}}}$. Formally $W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)$. We will think of $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}}$ as a matrix with each row containing a codeword.

- The algorithm $\mathsf{Trace}_{\mathsf{FP}}$ takes an n-user codebook W and a word $w' \in \{0,1\}^{\ell_{\mathsf{FP}}}$ and returns an index $i \in [n]$. Formally, $i = \mathsf{Trace}_{\mathsf{FP}}(W, w')$.

Given a non-empty subset $S \subseteq [n]$ and a set of codewords $W_S = (w^{(i)})_{i \in S} \in \{0,1\}^{\ell_{\mathsf{FP}}}$, we define the set of *feasible codewords* to be

$$F(W_S) = \left\{ w' \in \{0,1\}^{\ell_{\mathsf{FP}}} \mid \forall j \in [\ell_{\mathsf{FP}}]\, \exists i \in S\ w'_j = w_j^{(i)} \right\}.$$

Informally, if all users in S have a 0 (resp. 1) in the j-th symbol of their codeword, then they must produce a word with 0 (resp. 1) as the j-th symbol. We also define the *critical positions* to be the set of indices for which this constraint is binding. That is,

$$\mathrm{Crit}(W_S) = \left\{ j \in [\ell_{\mathsf{FP}}] \mid \forall i, i' \in S\ w_j^{(i)} = w_j^{(i')} \right\}.$$

The security of a fingerprinting code asserts that an adversary who is given a subset W_S of the codewords should not be able to produce an element of $F(W_S)$ that does not trace to a user $i \in S$. More formally,

Definition 3.12 (Secure Fingerprinting Code)**.** Let $\varepsilon_{\mathsf{FP}} \colon \mathbb{N} \to [0,1]$ and $\ell_{\mathsf{FP}} \colon \mathbb{N} \to \mathbb{N}$ be functions. A pair of algorithms $(\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ is an $(\varepsilon_{\mathsf{FP}}, \ell_{\mathsf{FP}})$-*fingerprinting code* if $\mathsf{Gen}_{\mathsf{FP}}(1^n)$ outputs a

codebook $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}(n)}$, and furthermore, for every (possibly inefficient) algorithm $\mathcal{A}_{\mathsf{FP}}$, and every non-empty $S \subseteq [n]$,

$$\Pr_{W \leftarrow_{\mathsf{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)} [\mathcal{A}_{\mathsf{FP}}(W_S) \in F(W_S) \wedge \mathsf{Trace}_{\mathsf{FP}}(W, \mathcal{A}_{\mathsf{FP}}(W_S)) \notin S] \leq \varepsilon_{\mathsf{FP}}(n)$$

where the two executions of $\mathcal{A}_{\mathsf{FP}}$ are understood to be the same.

Tardos [86] gave a construction of fingerprinting codes of essentially optimal length, improving on the original construction of Boneh and Shaw [18]. As we mentioned in in Section 3.4.1, the simple construction we sketched was just out eventual traitor-tracing scheme instantiated with the Boneh-Shaw fingerprinting code. The construction and analysis of Tardos' fingerprinting code also follows the blueprint of the tracing algorithm that we sketched: construct a random set of codewords such that 1) any feasible codeword will have some significant correlation with at least one of the codewords, and 2) it is information-theoretically impossible (due to the randomness in the codewords) to find a feasible codeword that has significant correlation with a codeword that you haven't seen, regardless of which codewords you have seen.

Theorem 3.13 ([86])**.** *For every function $\varepsilon_{\mathsf{FP}} : \mathbb{N} \to [0,1]$, there exists an $(\varepsilon_{\mathsf{FP}}, O(n^2 \log(n/\varepsilon_{\mathsf{FP}})))$-fingerprinting code. In particular, there exists a $(o(1/n^2), O(n^2 \log n))$-fingerprinting code.*

3.4.4 The Traitor-Tracing Scheme

We are now ready to state the construction more formally. The key generation, encryption, and decryption algorithms are as we described in the sketch (Section 3.4.1), and stated below.

3.4.5 Security of Π_{TT}

In this section we will prove that out construction of $\Pi_{\mathsf{TT}} = (\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}}, \mathsf{Trace}_{\mathsf{TT}})$ is an $(n, \ell_{\mathsf{FP}}(n))$-secure traitor-tracing scheme. It can be verified from the specification of the scheme that it has the desired syntactic properties, that it generates $n(\kappa)$ user keys, and that the tracing algorithm makes $\ell_{\mathsf{FP}}(n(\kappa))$ non-adaptive queries to its oracle.

Now we show how an available pirate decoder for this scheme can be traced. As in the sketch (Section 3.4.1), we want to generate a set of ciphertexts that different users decrypt in different ways. Specifically, given a fingerprinting code $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}}$ (represented as a matrix with

Let an encryption $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a function $n \colon \mathbb{N} \to \mathbb{N}$ be parameters of the scheme. Assume that $n(\kappa) \leq 2^{\kappa/2}$ for every $\kappa \in \mathbb{N}$

$\mathsf{Gen}_{\mathsf{TT}}(1^\kappa)$:

For: every user $i = 1, \ldots, n(\kappa)$

Let $\overline{sk}^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^{\kappa/2})$

Let $sk^{(i)} = (\overline{sk}^{(i)}, i)$ (padded with zeros to have length exactly κ).

Output: $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)})$

(We will sometimes use $sk^{(i)}$ and $\overline{sk}^{(i)}$ interchangeably)

$\mathsf{Enc}_{\mathsf{TT}}(sk^{(1)}, \ldots, sk^{(n)}, b)$:

For every user i, let $c^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, b)$

Output: $c = (c^{(1)}, \ldots, c^{(n)})$

$\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c)$:

Output: $\widehat{b} = \mathsf{Dec}(sk^{(i)}, c^{(i)})$

Figure 4: The algorithms $(\mathsf{Gen}_{\mathsf{TT}}, \mathsf{Enc}_{\mathsf{TT}}, \mathsf{Dec}_{\mathsf{TT}})$ for Π_{TT}.

$w^{(i)}$ in the i-th row), we want to generate a set of ciphertexts $c_1, \ldots, c_{\ell_{\mathsf{FP}}}$, such that user i, if she decrypts as intended using $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, \cdot)$, will decrypt c_j to $w_j^{(i)}$. That is, $\mathsf{Dec}_{\mathsf{TT}}(sk^{(i)}, c_j) = w_j^{(i)}$. $\mathsf{Trace}_{\mathsf{TT}}$ will query the pirate decoder on these ciphertexts, treat these responses as a word w', run the tracing algorithm for the fingerprinting code on w', and use the output of $\mathsf{Trace}_{\mathsf{FP}}$ as its own output.

If $\mathcal{P}$ is available, its output will be a feasible codeword for W_S. To see this, recall that if every user $i \in S$ decrypts c_j to the same bit, then an available pirate decoder $\mathcal{P}(\vec{sk}_S, \cdot)$, decrypts c_j to that bit. However, the critical positions of W_S are exactly those for which every user $i \in S$ has the same symbol in position j. Thus, the codeword returned by the pirate is feasible, and the fingerprinting code's tracing algorithm can identify a user in S.

The catch in this argument is that $\mathsf{TrEnc}_{\mathsf{TT}}$ takes all of W as input, however an attacker for the fingerprinting code is only allowed to see W_S, and thus cannot simulate $\mathsf{TrEnc}_{\mathsf{TT}}$ in a security

The tracing algorithm for Π_{TT} and the subroutine $\mathsf{TrEnc}_{\mathsf{TT}}$. Let a length $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$ fingerprinting code $\Pi_{\mathsf{FP}} = (\mathsf{Gen}_{\mathsf{FP}}, \mathsf{Trace}_{\mathsf{FP}})$ be a parameter of the scheme and let $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the encryption scheme used above.

$\mathsf{TrEnc}_{\mathsf{TT}}(sk^{(1)}, \ldots, sk^{(n)}, W)$:

Let $n \times k$ be the dimensions of W

For: every $i \in [n], j \in [k]$

Let $c_j^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, W_{i,j})$

For: every $j \in [k]$

Let $c_j = (c_j^{(1)}, \ldots, c_j^{(n)})$

Output: $c_1, \ldots, c_k$

(Notice that $\mathsf{Dec}(sk^{(i)}, c_j^{(i)}) = W_{i,j}$)

$\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}}(\vec{sk})$:

Let n be the number of user keys and $\ell_{\mathsf{FP}} = \ell_{\mathsf{FP}}(n)$

Let $W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)$

Let $\widehat{b}_1, \ldots, \widehat{b}_{\ell_{\mathsf{FP}}} \leftarrow_{\mathrm{R}} \mathcal{P}(\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W))$ and let $w' = \widehat{b}_1 \| \ldots \| \widehat{b}_{\ell_{\mathsf{FP}}}$

Output: $i \leftarrow_{\mathrm{R}} \mathsf{Trace}_{\mathsf{FP}}(W, w')$

Figure 5: The algorithm $\mathsf{Trace}_{\mathsf{TT}}$ for Π_{TT}

reduction. However, if $\mathcal{P}$ only has keys $\vec{sk}_S$, and $i \notin S$, then an efficient $\mathcal{P}$ cannot decrypt the i-th component of a ciphertext $c = (c^{(1)}, \ldots, c^{(n)})$. But these are the only components that depend on $w^{(i)}$. So $w^{(i)}$ is computationally hidden from $\mathcal{P}$ anyway, and we could replace that codeword with a string of zeros without significantly affecting the success probability of $\mathcal{P}$. Formalizing this intuition will yield a valid attacker for the fingerprinting code, and obtain a contradiction.

Theorem 3.14 (From Encryption to Traitor-Tracing)**.** *Let Π_{Enc} be an $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, T_{\mathsf{Enc}})$-secure encryption scheme, and Π_{FP} be a $(\varepsilon_{\mathsf{FP}}, \ell_{\mathsf{FP}})$-fingerprinting code, Π_{FP}. Let $n, k_{\mathsf{TT}} \colon \mathbb{N} \to \mathbb{N}$ be any functions such that for every $\kappa \in \mathbb{N}$, $n(\kappa) \leq 2^{\kappa/2}$ and*

1. *the encryption scheme and fingerprinting code have sufficiently strong security,*

$$n(\kappa) \cdot \varepsilon_{\mathsf{Enc}}(\kappa) + \varepsilon_{\mathsf{FP}}(n(\kappa)) = o\left(\frac{1}{n(\kappa)^2}\right),$$

2. *the encryption scheme is secure for sufficiently many queries,*

$$k_{\mathsf{Enc}}(\kappa) \geq k_{\mathsf{TT}}(\kappa) = \ell_{\mathsf{FP}}(n(\kappa)),$$

3. *the encryption scheme is secure against adversaries whose running time is as long as the pirate decoder's, for every $a > 0$,*

$$T_{\mathsf{Enc}}(\kappa/2, k_{\mathsf{TT}}(\kappa)) \geq (\kappa + n(\kappa) + k_{\mathsf{TT}}(\kappa))^a.$$

Then Π_{TT} instantiated with Π_{Enc} and Π_{FP} is an (n, k_{TT})-traitor-tracing scheme.

Proof. Suppose there exists a $\mathrm{poly}(\kappa, n(\kappa), k_{\mathsf{TT}}(\kappa))$-time pirate decoder $\mathcal{P}$ that violates the security of Π_{TT}. That is, for every $\kappa \in \mathbb{N}$, there exists $S = S(\kappa) \subseteq [n(\kappa)]$, $|S| \geq n(\kappa) - 1$, such that

$$\Pr_{\vec{sk} \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^\kappa)} \left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}(\vec{sk}_{S(\kappa)}, \cdot)}(\vec{sk}) \notin S \right] = \Omega\left(\frac{1}{n(\kappa)}\right)$$

where the probability is also taken over the coins of $\mathcal{P}$ and $\mathsf{Trace}_{\mathsf{TT}}$. Since there are only $n(\kappa)$ such sets, for a randomly chosen $i \leftarrow_{\mathrm{R}} [n(\kappa)]$, we have

$$\Pr_{\substack{\vec{sk} \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^\kappa) \\ i \leftarrow_{\mathrm{R}} [n(\kappa)]}} \left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}(\vec{sk}_{S_{-i}}, \cdot)}(\vec{sk}) \notin S \right] = \Omega\left(\frac{1}{n(\kappa)^2}\right).$$

Both of these probabilities are also taken over the coins of $\mathcal{P}$ and $\mathsf{Trace}_{\mathsf{TT}}$. We will show that such a pirate decoder must either violate the security of the encryption scheme or violate the security of the fingerprinting code.

Given a matrix $W \in \{0,1\}^{(n) \times \ell_{\mathsf{FP}}(n)}$, we define $W_{-i} \in \{0,1\}^{(n-1) \times \ell_{\mathsf{FP}}}$ to be W with the i-th codeword removed and $\widetilde{W}_{-i} \in \{0,1\}^{n \times \ell_{\mathsf{FP}}(n)}$ to be W with the i-th codeword replaced with $\vec{0}^{\ell_{\mathsf{FP}}(n)}$. We also use S_{-i} as a shorthand for $[n] \setminus \{i\}$

Consider the following algorithm $\mathcal{A}_{\mathsf{FP}}^{\mathcal{P}}$

Since the fingerprinting code is secure, for a randomly chosen $i \leftarrow_{\mathrm{R}} [n]$ (in fact, for every $i \in [n]$),

$$\Pr_{\substack{W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n) \\ i \leftarrow_{\mathrm{R}} [n]}} \left[\mathcal{A}_{\mathsf{FP}}^{\mathcal{P}}(S_{-i}, W_{-i}) \in F(W_{-i}) \wedge \mathsf{Trace}_{\mathsf{FP}}(W, \mathcal{A}_{\mathsf{FP}}^{\mathcal{P}}(S_{-i}, W_{-i})) = i \right] \leq \varepsilon_{\mathsf{FP}}(n) \qquad (3.3)$$

$\mathcal{A}^{\mathcal{P}}_{\mathsf{FP}}(S_{-i}, W_{-i})$:
Let n be the number of users for the fingerprinting code and κ be such that $n(\kappa) = n$
Generate keys $\vec{sk} \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{TT}}(1^\kappa)$ and ciphertexts $(c_1, \ldots, c_{\ell_{\mathsf{FP}}}) \leftarrow_{\mathrm{R}} \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, \widetilde{W}_{-i})$
Output $w' = (\widehat{b}_1, \ldots, \widehat{b}_{\ell_{\mathsf{FP}}}) \leftarrow_{\mathrm{R}} \mathcal{P}(\vec{sk}_{-i}, c_1, \ldots, c_{\ell_{\mathsf{FP}}})$
(Note that $\widetilde{W}_{-i}$ is just W_{-i} with a row of zeros added, so the attacker is well-defined.)

Figure 6: The fingerprinting security adversary.

This condition could hold simply because $\mathcal{A}_{\mathsf{FP}}$ outputs an infeasible codeword with high probability, not because we are successfully tracing a user in S. The next claim states that if $\mathcal{P}$ is an available pirate decoder, then this is not the case.

Claim 3.15. *Let $k_{\mathsf{TT}} = k_{\mathsf{TT}}(\kappa) = \ell_{\mathsf{FP}}(n(\kappa))$ for every $\kappa \in \mathbb{N}$. If $\mathcal{P}$ is a k_{TT}-available pirate decoder, then for every $\kappa \in \mathbb{N}$, every $i \in [n(\kappa)]$, and every $W \in \{0,1\}^{n \times \ell_{\mathsf{FP}}(n)}$ (for $n = n(\kappa)$)*

$$\Pr\left[\mathcal{A}^{\mathcal{P}}_{\mathsf{FP}}(S_{-i}, W_{-i}) \notin F(W_{-i})\right] = o\left(\frac{1}{n(\kappa)^2}\right)$$

Proof of Claim 3.15. If $\mathcal{P}$ is k_{TT}-useful, then, by definition, for every $\vec{sk} = (sk^{(1)}, \ldots, sk^{(n)})$, every $i \subseteq [n]$, and every $c_1, \ldots, c_{k_{\mathsf{TT}}}$, if every user $i' \neq i$ decrypts some c_j to the same bit b_j, then so does $\mathcal{P}(\vec{sk}_{-i}, \cdot)$ (with high probability). That is, for $\widehat{b}_1, \ldots, \widehat{b}_{k_{\mathsf{TT}}} \leftarrow_{\mathrm{R}} \mathcal{P}(\vec{sk}_{-i}, c_1, \ldots, c_{k_{\mathsf{TT}}})$,

$$\Pr\left[\exists j \in [k_{\mathsf{TT}}], b \in \{0,1\}\left(\left(\forall i' \neq i, \mathsf{Dec}_{\mathsf{TT}}(sk^{(i')}, c_j) = b\right) \wedge \left(\widehat{b}_j \neq b\right)\right)\right] = o\left(\frac{1}{n(\kappa)^2}\right) \quad (3.4)$$

Consider any critical position $j \in \mathrm{Crit}(W_{-i})$. These are the positions for which every user $i' \neq i$ has the same bit $w_j^{(i')} = b_j$. It's easy to see from the definition of $\mathsf{TrEnc}_{\mathsf{TT}}$ (and the correctness of Π_{Enc}) that if $c_1, \ldots, c_{k_{\mathsf{TT}}} \leftarrow_{\mathrm{R}} \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, \widetilde{W}_{-i})$ then every user $i' \neq i$ will decrypt c_j to b_j. Thus, with probability close to 1, for every critical position j, the j-th output of $\mathcal{P}(\vec{sk}_{-i}, c_1, \ldots, c_{k_{\mathsf{TT}}})$ will be equal to b_j, which implies $w' = (\widehat{b}_1, \ldots, \widehat{b}_{\ell_{\mathsf{FP}}})$ is feasible. □

Since $\mathcal{P}$ outputs feasible codewords with high probability, we obtain

$$\Pr_{\substack{W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n) \\ i \leftarrow_{\mathrm{R}} [n]}}\left[\mathsf{Trace}_{\mathsf{FP}}(W, \mathcal{A}^{\mathcal{P}}_{\mathsf{FP}}(S_{-i}, W_{-i})) = i\right] \leq \varepsilon_{\mathsf{FP}}(n(\kappa)) + o\left(\frac{1}{n(\kappa)^2}\right) \quad (3.5)$$

by combining the previous claim with (3.3).

There are only two differences between the success of the pirate decoder in fooling $\mathsf{Trace}_{\mathsf{TT}}$ and the success of the fingerprinting adversary in fooling $\mathsf{Trace}_{\mathsf{FP}}$ (in the experiment described in (3.5)): The first is that in the traitor-tracing security condition, $\mathcal{P}$ is given $\vec{sk}_{-i}$ for a fixed $i \in [n]$, whereas the fingerprinting adversary is given W_{-i} for a random $i \leftarrow_{\mathrm{R}} [n]$. This difference only affects the error by a factor of n. That is, for every $i \in [n]$

$$\Pr\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}(\vec{sk}_{-i},\cdot)}(\vec{sk}) = i\right] \leq n \cdot \Pr_{i \leftarrow_{\mathrm{R}}[n]}\left[\mathsf{Trace}_{\mathsf{TT}}^{\mathcal{P}(\vec{sk}_{-i},\cdot)}(\vec{sk}) = i\right]$$

The second difference is that in $\mathsf{Trace}_{\mathsf{TT}}$, the ciphertexts given to the pirate are generated by $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W)$ whereas in $\mathcal{A}_{\mathsf{FP}}$ the ciphertexts are generated by $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, \widetilde{W}_{-i})$. But these ciphertexts only differ in the i-th component, and $sk^{(i)}$ is unknown to $\mathcal{P}$, so this does not affect the behavior of the pirate decoder significantly. This fact is established in the following claim.

Claim 3.16. *If Π_{Enc} is $(\varepsilon_{\mathsf{Enc}}, k_{\mathsf{Enc}}, T_{\mathsf{Enc}})$-secure for $k_{\mathsf{Enc}}, T_{\mathsf{Enc}}$ as in the statement of the Theorem, then for every $\mathrm{poly}(\kappa, n(\kappa), k_{\mathsf{TT}}(\kappa))$ pirate decoder $\mathcal{P}$,*

$$\left| \Pr_{\substack{W \leftarrow_R \mathsf{Gen}_{\mathsf{FP}}(1^n) \\ \vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}, i \leftarrow_R [n]}} \left[\mathsf{Trace}_{\mathsf{FP}}(W, \mathcal{P}(\vec{sk}_{-i}, \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W))) = i\right] \right.$$
$$\left. - \Pr_{\substack{W \leftarrow_R \mathsf{Gen}_{\mathsf{FP}}(1^n) \\ \vec{sk} \leftarrow_R \mathsf{Gen}_{\mathsf{TT}}, i \leftarrow_R [n]}} \left[\mathsf{Trace}_{\mathsf{FP}}(W, \mathcal{P}(\vec{sk}_{-i}, \mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, \widetilde{W}_{-i}))) = i\right] \right| \leq \varepsilon_{\mathsf{Enc}}(\kappa)$$

Proof of Claim 3.16. Let $\Pi_{\mathsf{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the encryption scheme. The main observation required to prove the claim is that the two experiments we want to relate can both be simulated without $sk^{(i)}$, given challenges for the encryption scheme (Definition 3.10). Fix a codebook $W \leftarrow_{\mathrm{R}} \mathsf{Gen}_{\mathsf{FP}}(1^n)$. Now consider two distributions on ciphertexts (of Π_{Enc}): In either case, generate a random key $sk^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)$

- In the first case $c_1^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, w_1^{(i)}), \ldots, c_{\ell_{\mathsf{FP}}}^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, w_{\ell_{\mathsf{FP}}}^{(i)})$
- In the second case $sk^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa)$ and $c_1^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, 0), \ldots, c_{\ell_{\mathsf{FP}}}^{(i)} \leftarrow_{\mathrm{R}} \mathsf{Enc}(sk^{(i)}, 0)$

Suppose we receive a set of ℓ_{FP} ciphertexts from one of these two distributions. Note that $\mathsf{Gen}_{\mathsf{TT}}$ chooses keys for each user independently, and $\mathsf{TrEnc}_{\mathsf{TT}}$ generates ciphertext components for each user independently. So we can generate keys $\vec{sk}_{-i}$, and ciphertext components for users other than i independently, and use the challenge ciphertexts in place of the ciphertext components for user i, without knowing $sk^{(i)}$. Suppose we simulate $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W)$ in this way. Notice that if the

challenge ciphertexts come from the first distribution, then simulated ciphertexts will be distributed exactly as in $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, W)$, and if the challenge ciphertexts come from the second distribution, then the simulated ciphertexts will be distributed exactly as in $\mathsf{TrEnc}_{\mathsf{TT}}(\vec{sk}, \widetilde{W}_{-i})$. But, if the claim were false, then we would have found an adversary for the encryption scheme that can distinguish between the two distributions with advantage greater than $\varepsilon_{\mathsf{Enc}}(\kappa)$. It is easy to see that if the pirate decoder is efficient, then so will the adversary for the encryption scheme (since $\mathsf{Trace}_{\mathsf{FP}}, \mathsf{Gen}, \mathsf{Enc}$ are all assumed to be efficient. We conclude that if the claim is false, then $\mathcal{A}_{\mathsf{Enc}}$ violates the security of Π_{Enc}. □

We now complete the proof of the theorem by combining Equation (3.5) and Claim 3.16.

□

3.4.6 Decryption Function Family of Π_{TT}

Recall that the two goals of constructing a new traitor-tracing scheme were to trace stateful pirates and to reduce the complexity of decryption. We addressed tracing of stateful pirates in the previous section, and now we turn to the complexity of decryption. We do so by instantiating the traitor-tracing scheme with various encryption schemes and making two observations: 1) The type of encryption schemes we require are sufficiently weak that there already exist plausible candidates with a very simple decryption operation, and 2) Decryption for the traitor-tracing scheme is not much more complex than decryption for the underlying encryption scheme. We summarize the second point with the following simple lemma.

Lemma 3.17 (Decryption Function Family for Π_{TT})**.** *Let Π_{TT} be as defined, with Π_{Enc} as its underlying encryption scheme. Let $(\overline{sk}, i) = sk \in \{0,1\}^{\kappa}$ and $c = (c^{(1)}, \ldots, c^{(n)}) \in \mathcal{C}^{(\kappa)}$ be any user key and ciphertext for Π_{TT}. Then*

$$\mathsf{Dec}_{\mathsf{TT},c}(sk) = \mathsf{Dec}_{\mathsf{TT},c}(\overline{sk}, i) = \bigvee_{i' \in [n]} \left(\mathbf{1}_{i'}(i) \wedge \mathsf{Dec}_{c^{(i')}}(\overline{sk}) \right)$$

Here, the function $\mathbf{1}_x(y)$ takes the value 1 if $y = x$ and 0 otherwise. The lemma follows directly from the construction of $\mathsf{Dec}_{\mathsf{TT}}$. Also note that the function $\mathbf{1}_{i'} \colon \{0,1\}^{\lceil \log n \rceil} \to \{0,1\}$ is just a conjunction of $\lceil \log n \rceil$ bits (a single gate of fan-in $O(\log n)$), and we need to compute n of these functions. In addition to computing $\mathbf{1}_{i'}$ and $\mathsf{Dec}_{c^{(i')}}$, there are n conjunctions and a single outer

disjunction. Thus we add an additional $n+1$ gates, compute decryption n times, and increase the depth by 2. Hence, an intuitive summary of the lemma is that if Dec can be implemented by circuits of size s and depth h, $\mathsf{Dec}_{\mathsf{TT}}$ can be implemented by circuits of size $n \cdot (s + O(\log n)) = \widetilde{O}(ns)$ and depth $h+2$. This summary will be precise enough to state our main results.

By combining Lemma 3.17 with Theorem 3.14, we easily obtain the following corollary.

Corollary 3.18 (One-way Functions Imply Traitor-Tracing w/ Poly-Time Decryption)**.** *Let $n = n(\kappa)$ be any polynomial in κ. Assuming the existence of (non-uniformly secure) one-way functions, there exists an $(n, \widetilde{O}(n^2))$-secure traitor-tracing scheme with decryption function family $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}},\kappa}$ consisting only of circuits of size $\mathrm{poly}(\kappa)$*

Proof. The existence of one-way functions implies the existence of an encryption scheme Π_{Enc} that is $(1/\kappa^a, \kappa^a, \kappa^a)$-secure for every constant $a > 0$ and sufficiently large κ with decryption function $\mathcal{Q}_{\mathsf{Dec},\kappa}$ consisting only of circuits of size $t(\kappa) = \mathrm{poly}(\kappa)$ for every $\kappa \in \mathbb{N}$. From Lemma 3.17, it is easy to see that if Π_{TT} uses Π_{Enc} as its encryption scheme, then $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}},\kappa}$ consists only of circuits of size $\widetilde{O}(n(\kappa)t(\kappa/2)) = \mathrm{poly}(\kappa)$. □

Theorem 6.1 in the introduction follows by combining Theorem 3.6 with Corollary 3.18.

We will now consider the possibility of constructing a traitor-tracing scheme where the decryption functionality can be implemented by circuits of constant depth, and thus obtaining hardness results for generic sanitizers that are efficient for constant-depth queries (Theorem 6.3). First, we summarize our observation that the traitor-tracing scheme almost preserves the depth of the decryption function.

Corollary 3.19 (Encryption with Constant-Depth Decryption Impies Traitor-Tracing w/ Constant-Depth Decryption)**.** *Let $n = n(\kappa)$ be any polynomial in κ. If there exists an encryption scheme, $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, that is $(o(1/n^2), \omega(n^4), n^a)$-secure for every $a > 0$ and has decryption family $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}}$ consisting of circuits of size $\mathrm{poly}(\kappa)$ and depth h, then there exists a $(n, \widetilde{O}(n^2))$-secure traitor-tracing scheme with decryption function family $\mathcal{Q}^{(\kappa)}_{\mathsf{Dec}_{\mathsf{TT}}}$ consisting of circuits of size $\widetilde{O}(n) \cdot \mathrm{poly}(\kappa)$ and depth $h+2$.*

The corollary is clear from Lemma 3.17 and Theorem 3.14. The corollary is not interesting without an encryption scheme that can be decrypted by constant-depth circuits. However, we observe that such a scheme (meeting our relaxed security criteria) can be constructed from a sufficiently good *local pseudorandom generator (PRG)*. A recent result of Applebaum [3] gave the

first plausible candidate construction of a local PRG for the range of parameters we need, giving plausibility to the assumption that such PRGs (and, as we show, traitor-tracing schemes with constant-depth decryption) exist. We note that local PRGs actually imply encryption schemes with local decryption, which is stronger than just constant-depth decryption. Although it may be significantly easier to construct encryption schemes that only have constant-depth decryption, we are not aware of any other ways of constructing such a scheme.

Definition 3.20 (Local Pseudorandom Generator)**.** An algorithm $\mathsf{G}\colon \{0,1\}^{\kappa} \to \{0,1\}^{s_{\mathsf{PRG}}(\kappa)}$ is a $\varepsilon_{\mathsf{PRG}}$-pseudorandom generator if it is efficient ($\mathrm{poly}(\kappa)$-time) and for every $\mathrm{poly}(s_{\mathsf{PRG}}(\kappa))$-time adversary $\mathcal{A}_{\mathsf{PRG}}$

$$\left|\Pr\left[\mathcal{A}_{\mathsf{PRG}}(\mathsf{G}(U_\kappa)) = 1\right] - \Pr\left[\mathcal{A}_{\mathsf{PRG}}(U_{s_{\mathsf{PRG}}(\kappa)}) = 1\right]\right| \leq \varepsilon_{\mathsf{PRG}}(\kappa)$$

If, in addition, if each bit of the output depends only on some set of L bits of the input, then G is a $(\varepsilon_{\mathsf{PRG}}, L)$-local pseudorandom generator.

It is a well known result in Cryptography that pseudorandom generators imply encryption schemes satisfying Definition 3.10 (for certain ranges of parameters). We will use a particular construction whose decryption can be computed in constant-depth whenever the underlying PRG is locally-computable (or, more generally, computable by constant-depth circuits). The construction is the standard "computational one-time pad", however we give a construction to verify that the decryption can be computed by constant-depth circuits.

$\mathsf{Gen}(1^{\kappa})$:

Let $s \leftarrow_{\mathrm{R}} \{0,1\}^{\kappa}$ and output $sk = s$

$\mathsf{Enc}(sk, b)$:

Let $r \leftarrow_{\mathrm{R}} \{1, 2, \ldots, s_{\mathsf{PRG}}(\kappa)\}$ and output $c = (r, \mathsf{G}(sk)_r \oplus b)$

$\mathsf{Dec}(sk, c)$:

Let $(r', b') = c$ and output: $b = \mathsf{G}(sk)_r \oplus b'$

Figure 7: An encryption scheme Π_{LocalEnc} that can be decrypted in constant depth.

Lemma 3.21 (Local PRGs $\to$ Encryption)**.** *If there exists a* $(\varepsilon_{\mathsf{PRG}}(\kappa), L)$*-local pseudorandom generator* $\mathsf{G}\colon \{0,1\}^{\kappa} \to \{0,1\}^{s_{\mathsf{PRG}}(\kappa)}$*, then there exists an* $(\varepsilon_{\mathsf{Enc}} = \varepsilon_{\mathsf{PRG}} + k_{\mathsf{Enc}}^2/s_{\mathsf{PRG}}, k_{\mathsf{Enc}})$*-Secure Encryption Scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with decryption function family* $\mathcal{Q}_{\mathsf{Dec},\kappa}$ *consisting of circuits of size* $\mathrm{poly}(\kappa)$ *and depth* 4*.*

Proof. The security follows from standard arguments: If we choose a random $s \leftarrow_{\mathrm{R}} \{0,1\}^{\kappa}$, then $\mathsf{G}(s)$ is indistinguishable from uniform up to error $\varepsilon_{\mathsf{PRG}}$. If we generate k_{Enc} encryptions with key s, and no two encryptions use the same choice of r, then the output is indistinguishable from encryptions using uniform random bits in place of $\mathsf{G}(s)$. If we use uniform random bits in place of G, then the message is information-theoretically hidden. The probability that no two encryptions out of k_{Enc} use the same choice of r is at most $k_{\mathsf{Enc}}^2/s_{\mathsf{PRG}}$, so we lose this term in the security of the encryption scheme.

Let $\mathbf{1}_i(j)$ be the indicator variable for the condition $j = i$. For every $c = (r, b) \in \mathcal{C}$, we can write

$$\mathsf{Dec}_{(r,b)}(s) = \bigvee_{i \in [s_{\mathsf{PRG}}(\kappa)]} \left(\mathbf{1}_i(r) \wedge (\mathsf{G}_i(s) \oplus b)\right).$$

Observe that, since G_i is a function of L bits of the input, it can be computed by a size-2^L DNF (depth-2 circuit), thus $\mathsf{G}_i(s) \oplus b$ can be computed by a size $2^L + 1$, depth-3 circuit. The indicator $\mathbf{1}_i$ can be computed by a conjunction of $\lceil \log_2 s_{\mathsf{PRG}}(\kappa) \rceil$ bits, which is a size-$\lceil \log_2 s_{\mathsf{PRG}}(\kappa) \rceil$, depth-1 circuit. The outer disjunction increases the depth by one level and the size by 1. Putting it all together, we have shown that $\mathsf{Dec}_{r,b}(s)$ can be computed by depth-4 circuits of size $\widetilde{O}(2^L s_{\mathsf{PRG}}(\kappa)) = \mathrm{poly}(s_{\mathsf{PRG}}(\kappa))$. □

Combining Corollary 3.19 with Lemma 3.21 easily yields the following corollary.

Corollary 3.22 (Local Pseudorandom Generators Imply traitor-tracing w/ AC^0 Decryption)**.** *Let* $n = n(\kappa)$ *be any polynomial in* κ*. Assuming the existence of a* $(o(1/n^2), n^7, L)$*-local pseudorandom generator for some constant* $L \in \mathbb{N}$*, there exists an* $(n, \widetilde{O}(n^2))$*-secure traitor-tracing scheme with decryption function family* $\mathcal{Q}_{\mathsf{Dec}_{\mathsf{TT}},\kappa}$ *consisting of circuits of size* $\widetilde{O}(n) \cdot \mathrm{poly}(\kappa)$ *and depth* 6*.*

Theorem 6.3 in the introduction follows by combining Theorem 3.6 with Corollary 3.22.

Chapter 4

The Hardness of Generating Private Synthetic Data

In Chapter 3 we saw that if we want efficient sanitizers for counting queries that provide better utility than the Laplace mechanism, then we have to do something more than just restrict the *number* of queries (as answering $n^{2+o(1)}$ queries is hard, by Theorem 3.1). In this chapter we take a different approach and consider restricting the *complexity* of the queries. In particular, we focus on a restricted family of counting queries, called *marginal queries*. Recall from the introduction that a marginal query is specified by a set $S \subseteq [d]$ and a pattern $t \in \{0,1\}^{|S|}$. The query asks, "What fraction of the individual records in D has each of the attributes $j \in S$ set to t_j?" A natural restriction is to the set of *k-way marginal queries*, which are marginal queries specified by sets of size at most k. The set of answers to all k-way marginal queries is alternatively known as the "k-way contingency table" of the database. Contingency tables are a workhorse of categorical data analysis, as they are easy to interpret and are sufficient statistics for many popular probabilistic models. For example, the set of pairwise correlations between different binary attributes—the covariance matrix—is equivalent to the 2-way contingency table. Marginal queries are also sometimes known as "conjunction queries" in the differential privacy literature, since a marginal query is simply a counting query in which the predicate q is a conjunction on the bits of it input.

In this chapter, we focus on *one-shot sanitizers* for k-way marginal queries. Recall that a one-shot sanitizer computes and releases a single differentially private "summary" of the database that enables others to determine accurate answers to a large class of queries. What form should this

summary take? The most appealing form would be a *synthetic database* (defined in Chapter 2), which is a new database $\widehat{D} = \mathcal{M}(D)$ whose rows are "fake", but come from the same universe as those of D and are guaranteed to share many statistics with those of D (up to some accuracy). In particular, we would like the synthetic database to yield answers to k-way marginal queries that are approximately the same as those given by the input database. Some advantages of synthetic data are that it can be easily understood by humans, and statistical software can be run directly on it without modification.

The first result on producing differentially private synthetic data came in the work of Barak et al. [8]. Given a database D consisting of n rows from $\{0,1\}^d$, they show how to construct a differentially private synthetic database $\widehat{D}$, also of n rows from $\{0,1\}^d$, in which the answer to every marginal query, is approximately preserved up to an additive error of $2^{O(d)}/n$. The running time of their algorithm is $\mathrm{poly}(n, 2^d)$, which is feasible for small values of d. They pose as an open problem whether the running time of their algorithm can be improved for the case where we only want to preserve the *k-way marginals* for small k (e.g. $k = 2$). Indeed, the number of k-way marginal queries is only $d^{\Theta(k)}$ (when $k \ll d$), and we can produce differentially private estimates for all of these queries in time $\mathrm{poly}(n, d^k)$ with error $d^{\Theta(k)}/n$, using the Laplace mechanism (Lemma 2.9). Moreover, a version of the Barak et al. algorithm [8] can ensure that even these noisy answers are consistent with a real database.[7]

As we have discussed, there are more powerful techniques than the Laplace mechanism for answering counting queries. Indeed, all of these techniques generate synthetic data, either explicitly or as a byproduct of their design. For every class $\mathcal{Q} = \{q : \{0,1\}^d \to \{0,1\}\}$ of predicates, the private multiplicative weights algorithm (Lemma 2.16) yields a differentially private one-shot sanitizer $\mathcal{M}$ that produces a synthetic database $\widehat{D} = \mathcal{M}(D)$ such that all counting queries corresponding to predicates in $\mathcal{Q}$ are preserved to within an accuracy of $\widetilde{O}(d^{1/4}\sqrt{\log|\mathcal{Q}|}/n^{1/2})$, with high probability. In particular, with $n = \mathrm{poly}(d)$, the synthetic data can provide simultaneous accuracy for an *exponential-sized* family of queries (e.g. $|\mathcal{Q}| = 2^d$). Indeed, nearly every technique we are aware of for answering a large number of counting queries crucially relies on synthetic data as part of its design.[8] Unfortunately, the running time of every such known algorithm is at least 2^d.

[7] Technically, this "real database" may assign fractional weight to some rows.

[8] A notable exception is the new algorithm of Nikolov, Talwar, and Zheng [70], which does not explicitly maintain a synthetic database as part of its state.

Dwork et al. [32] gave evidence that the large running time of the these latter algorithms is inherent. Specifically, assuming the existence of one-way functions, they exhibit an efficiently computable family $\mathcal{Q}$ of predicates (e.g. consisting of circuits of size d^2) for which it is infeasible to produce a differentially private synthetic database preserving the counting queries corresponding to $\mathcal{Q}$ (for databases of any $n = \mathrm{poly}(d)$ number of rows). However, these results left open the possibility that for *natural* families of counting queries (e.g. k-way marginal queries), producing a differentially private synthetic database (or non-synthetic summarization) can be done efficiently. Indeed, one may have gained optimism from a comparison to the early history of computational learning theory, where one-way functions were used to show hardness of learning arbitrary efficiently computable concepts in computational learning theory but natural subclasses (like conjunctions) were found to be learnable [91].

4.1 Our Results and Techniques

We prove that it is infeasible to produce synthetic databases preserving even very simple counting queries, such as 2-way marginals:

Theorem 4.1. *Assuming the existence of one-way functions, there is a constant $\alpha > 0$ such that for any constant $\beta < 1$, there is no efficient one-shot sanitizer whose output is a synthetic database that is (α, β)-accurate for 2-way marginals.*

That is, for every polynomial p, there is no polynomial-time, differentially private algorithm $\mathcal{M}$ that takes a database $D \in (\{0,1\}^d)^{p(d)}$ and produces a synthetic database $\widehat{D} \in (\{0,1\}^d)^*$ such that $|q(D) - q(\widehat{D})| \leq \alpha$ for all 2-way marginals q.

Stated differently, there is no efficient, differentially private, one-shot sanitizer for the family of 2-way marginals whose summary is a synthetic database. In fact, our impossibility result extends from conjunctions of 2 literals to any family of constant arity predicates that contains a function depending on at least two variables, such as parities of 3 literals.

As mentioned earlier, all 2-way marginals *can* be easily summarized with non-synthetic data (by just adding noise to each of the $(2d)^2$ values). Thus, our result shows that requiring a synthetic database may severely constrain what sorts of differentially private data releases are possible. (Dwork et al. [32] also showed that there exists a $\mathrm{poly}(d)$-sized family of counting queries that

are hard to summarize with synthetic data, thereby separating synthetic data from non-synthetic data. Our contribution is to show that such a separation holds for a very simple and natural family of predicates, namely 2-way marginals.)

This separation between synthetic data and non-synthetic data seems analogous to the separations between proper and improper learning in computational learning theory [73, 36], where it is infeasible to learn certain concept classes if the output hypothesis is constrained to come from the same representation class as the concept, but it becomes feasible if we allow the output hypothesis to come from a different representation class. This analogy gives hope for designing efficient, differentially private algorithms that take a database and produce a compact summary of it that is not synthetic data but somehow can be used to accurately answer exponentially many questions about the original database (e.g. all marginals). The negative results of Dwork et al. [32] and those of Chapter 3 on non-synthetic data (assuming the existence of efficient traitor-tracing schemes) do not say anything about natural classes of counting queries, such as marginals. Indeed, in Chapters 5 and 6 we will design one-shot sanitizers for marginal queries that do not generate synthetic data and are faster than what can be achieved for arbitrary counting queries.

To bypass the complexity barrier stated in Theorem 4.1, it may not be necessary to introduce exotic data representations; some mild generalizations of synthetic data may suffice. For example, several recent algorithms [16, 78, 35] produce several synthetic databases, with the guarantee that the *median* answer over these databases is approximately accurate. More generally, we can consider summarizations of a database D that that consist of a collection $\widehat{D}$ of rows from the same universe as the original database, and where we estimate $q(D)$ by applying the predicate q to each row of $\widehat{D}$ and then aggregating the results via some aggregation function f. With standard synthetic data, f is simply the average, but we may instead allow f to take a median of averages, or apply an affine shift to the average. For such *relaxed synthetic data*, we prove the following results:

- There is a constant k such that counting queries corresponding to k-juntas (functions depending on at most k variables) *cannot* be accurately and privately summarized as relaxed synthetic data with a median-of-averages aggregator, or with a symmetric and monotone aggregator (that is independent of the predicate q being queried).

- For every constant k, counting queries corresponding to k-juntas *can* be accurately and privately summarized as relaxed synthetic data with an aggregator that applies an affine shift to

the average (where the shift does depend on the predicate being queried).

Our proof of Theorem 4.1 and our other negative results are obtained by combining the hard-to-sanitize databases of Dwork et al. [32] with PCP reductions. They construct a database consisting of valid message-signature pairs (m_i, σ_i) under a digital signature scheme, and argue that any differentially private sanitizer that preserves accuracy for the counting query associated with the signature verification predicate can be used to forge valid signatures. We replace each message-signature pair (m_i, σ_i) with a PCP encoding π_i that proves that (m_i, σ_i) satisfies the signature verification algorithm. We then argue that if accuracy is preserved for a large fraction of the (constant arity) constraints of the PCP verifier, then we can "decode" the PCP either to violate privacy (by recovering one of the original message-signature pairs) or to forge a signature (by producing a new message-signature pair).

We remark that error-correcting codes were already used in [32] for the purpose of producing a fixed polynomial-sized set of counting queries that can be used for all verification keys. Our observation is that by using *PCP encodings*, we can reduce not only the number of counting queries in consideration, but also their computational complexity.

Our proof has some unusual features among PCP-based hardness results:

- As far as we know, this is the first time that PCPs have been used in conjunction with cryptographic assumptions for a hardness result. (They have been used together for positive results regarding computationally sound proof systems [55, 64, 9].) It would be interesting to see if such a combination could be useful in, say, computational learning theory (where PCPs have been used for hardness of "proper" learning [2, 37] and cryptographic assumptions for hardness of representation-independent learning [91, 52]).

- While PCP-based inapproximability results are usually stated as Karp reductions, we actually need them to be *Levin* reductions—capturing that they are reductions between search problems, and not just decision problems. (Previously, this property has been used in the same results on computationally sound proofs mentioned above.)

4.2 Relationship with Hardness of Approximation

The objective of a privacy-preserving sanitizer is to reveal some properties of the underlying database without giving away enough information to reconstruct that database. This requirement has different implications for sanitizers that produce synthetic databases and those with arbitrary output.

The SuLQ framework of [15] is a well-studied and efficient technique for achieving (ϵ, δ)-differential privacy, with non-synthetic output. To get accurate, private output for a family of counting queries with predicates in $\mathcal{Q}$, we can release a vector of noisy counts $(q(D) + Z_q)_{q \in \mathcal{Q}}$ where the random variables $(Z_q)_{q \in \mathcal{Q}}$ are drawn independently from a distribution suitable for preserving privacy (e.g. a Laplace distribution with standard deviation $O(|\mathcal{Q}| / \epsilon n)$).

Consider the case of an n-row database D that contains satisfying assignments to a 3CNF formula φ, and suppose our concept class includes all disjunctions on three literals (or, equivalently, all conjunctions on three literals). Then the technique above releases a set of noisy counts that describes a database in which every clause of φ is satisfied by most of the rows of D. However, sanitizers with synthetic-database output are required to produce a database that consists of rows that satisfy most of the clauses of φ.

Because of the noise added to the output, the requirement of a synthetic database does not strictly force the sanitizer to find a satisfying assignment for the given 3CNF. However, it is known to be NP-hard to find even approximate satisfying assignments to 3CNF formulae. In our main result, Theorem 4.16, we will show that there exists a distribution over databases that is hard-to-sanitize with respect to synthetic data for any concept class that is sufficient to express a hard-to-approximate constraint satisfaction problem.

4.2.1 Hard to Approximate CSPs

We define a *constraint satisfaction problem* to be the following.

Definition 4.2 (Constraint Satisfaction Problem (CSP))**.** For a non-decreasing function $q = q(d) \leq d$, a *family of $q(d)$-CSPs*, denoted $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$, is a sequence of sets Γ_d of boolean predicates defined on up to $q(d)$ variables. We assume for convenience that $\Gamma_1 \subseteq \Gamma_2 \subseteq \ldots$. If $q(d)$ and Γ_d do not depend on d then we refer to Γ as a *fixed family of q-CSPs*.

For every $d \geq q(d)$, let $\mathcal{Q}_\Gamma^{(d)}$ be the class consisting of all predicates $q : \{0,1\}^d \to \mathbb{R}$ of the form $q(u_1, \ldots, u_d) = \gamma(u_{i_1}, \ldots, u_{i_{q(d)}})$ for some $\gamma \in \Gamma_d$ and $i_1, \ldots, i_{q(d)} \in [d]$. We call $\mathcal{Q}_\Gamma = \cup_{d=0}^{\infty} \mathcal{Q}_\Gamma^{(d)}$ the *class of constraints of* Γ. Finally, we say a multiset $\varphi \subseteq \mathcal{Q}_\Gamma^{(d)}$ is a *d-variable instance of* $\mathcal{Q}_\Gamma$ and each $\varphi_i \in \varphi$ is a *constraint of* φ.

We say that an assignment π *satisfies* the constraint φ_i if $\varphi_i(\pi) = 1$. For $\varphi = \{\varphi_1, \ldots, \varphi_m\}$, define

$$\mathrm{val}(\varphi, \pi) = \frac{1}{m} \sum_{i=1}^{m} \varphi_i(\pi) \qquad \text{and} \qquad \mathrm{val}(\varphi) = \max_{\pi \in \{0,1\}^d} \mathrm{val}(\varphi, \pi).$$

Our hardness results will apply to concept classes $\mathcal{Q}_\Gamma^{(d)}$ for CSP families Γ with certain additional properties. Specifically we define,

Definition 4.3 (Nice CSP)**.** A family $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ of $q(d)$-CSPs *nice* if

1. $q(d) = d^{1-\Omega(1)}$, and
2. for every $d \in \mathbb{N}$, there exists a non-constant predicate $\varphi^* : \{0,1\}^{q(d)} \to \{0,1\}$, and two assignments $u_0^*, u_1^* \in \{0,1\}^{q(d)}$ such that $\varphi^*(u_0) = 0$ and $\varphi^*(u_1) = 1$ can be found in time $\mathrm{poly}(d)$.

We note that any fixed family of q-CSP that contains a non-constant predicate is a nice CSP. Indeed, these CSPs (e.g. conjunctions of 2 literals) are the main application of interest for our results. However it will sometimes be useful to work with generalizations to nice CSPs with predicates of non-constant arity.

For our hardness result, we will need to consider a strong notion of hard constraint satisfaction problems, which is related to probabilistically checkable proofs. First we recall the standard notion of hardness of approximation under Karp reductions (stated for additive, rather than multiplicative approximation error).

Definition 4.4 (Inapproximability under Karp Reductions)**.** Let $\alpha, \gamma : \mathbb{N} \to [0,1]$ be functions. A family of CSPs $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ is *(α, γ)-hard-to-approximate under Karp reductions* if there exists a polynomial-time computable function R such that for every circuit C with $|C|$ gates and input size $\overline{d}$, if we set $\varphi_C = R(C) \subseteq \mathcal{Q}_\Gamma$, then

1. if C is satisfiable, then $\mathrm{val}(\varphi_C) \geq \gamma(d)$, and

2. if C is unsatisfiable, then $\mathrm{val}(\varphi_C) < \gamma(d) - \alpha(d)$.

For our hardness result, we will need a stronger notion of inapproximability, which says that we can efficiently transform satisfying assignments of C into solutions to φ_C of high value, and vice-versa. In order to make the statement of our hardness result more precise, we also want to put explicit bounds on both the input length of the instance produced by the reduction and the time required to transform assignments to C into solutions to φ_C and vice versa.

Definition 4.5 (Inapproximability under Levin Reductions)**.** Let $\alpha, \gamma : \mathbb{N} \to [0,1]$ and $L_{\mathrm{PCP}} : \mathbb{N} \to \mathbb{N}$ be functions. A family of CSPs $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ is (α, γ)*-hard-to-approximate under Levin reductions with length blowup* L_{PCP} if there exist polynomial-time computable functions R, Enc, Dec such that for every circuit C with $|C|$ gates and input of size $\overline{d}$

1. $\varphi_C = R(C) \subseteq \mathcal{Q}_\Gamma^{(d)}$ for $d = L_{\mathrm{PCP}}(|C|)$,
2. for every $u \in \{0,1\}^{\overline{d}}$ such that $C(u) = 1$, $\mathrm{val}(\varphi_C, Enc(u, C)) \geq \gamma(d)$,
3. for every $\pi \in \{0,1\}^d$ such that $\mathrm{val}(\varphi_C, \pi) \geq \gamma(d) - \alpha(d)$, $C(Dec(\pi, C)) = 1$,
4. for every $u \in \{0,1\}^{\overline{d}}$, $Dec(Enc(u, C)) = u$, and
5. Enc and Dec are both computable in time $\mathrm{poly}(|C|, d)$.

When we do not wish to specify the value γ we will simply say that the family Γ is α*-hard-to-approximate under Levin reductions with length blowup* L_{PCP} to indicate that there exists such a $\gamma \in (\alpha, 1]$. When we do not specify a length blowup L_{PCP}, then L_{PCP} is assumed to be a polynomial. If we drop the requirement that R is efficiently computable, then we say that Γ is (α, γ)*-hard-to-approximate under inefficient Levin reductions with length blowup* L_{PCP}. Finally, if $L_{\mathrm{PCP}}(s) \leq s^{1+o(1)}$ then we will write that Γ is (α, γ)*-hard-to-approximate under Levin reductions with nearly-linear length blowup*.

The notation Enc, Dec reflects the fact that we think of the set of assignments π such that $\mathrm{val}(\varphi_C, \pi) \geq \gamma$ as a sort of error-correcting code on the satisfying assignments to C. Any π with value close to γ can be decoded to a valid satisfying assignment.

Levin reductions are a stronger notion of reduction than Karp reductions. To see this, let Γ be α-hard-to-approximate under Levin reductions, and let R, Enc, Dec be the functions described in

Definition 4.5. We now argue that for every circuit C, the formula $\varphi_C = R(C)$ satisfies conditions 1 and 2 of Definition 4.4. Specifically, if there exists an assignment $u \in \{0,1\}^{\overline{d}}$ that satisfies C, then $Enc(u, C)$ satisfies at least a γ fraction of the constraints of φ_C. Conversely if any assignment $\pi \in \{0,1\}^d$ satisfies at least a $\gamma - \alpha$ fraction of the constraints of φ_C, then $Dec(\pi, C)$ is a satisfying assignment of C.

Variants of the PCP Theorem can be used to show that essentially every class of CSP is hard-to-approximate in this sense. Indeed, $\pi = Enc(u, C)$ corresponds to a "probabilistically checkable proof" that C is satisfiable: if we check a random constraint of $\varphi_C = R(C)$ (which requires reading only a few bits of π) then we will accept with probability at least γ. Conversely, if the above test passes for some string $\pi \in \{0,1\}^d$ with probability at least $\gamma - \alpha$, then C must be satisfiable (as $Dec(\pi, C)$ is a satisfying assignment to C).

We restrict to CSP's that are closed under negation as it suffices for our application.

Theorem 4.6 (variant of PCP Theorem)**.** *For every fixed family of CSPs Γ that is closed under negation and contains a function that depends on at least two variables, there is a constant $\alpha = \alpha(\Gamma) > 0$ such that Γ is α-hard to approximate under Levin reductions.*

Proof sketch. Hardness of approximation under Karp reductions follows directly from the classification theorems of Creignou [24] and Khanna et al. [53]. These theorems show that all CSPs are either α-hard under Karp reductions for some constant $\alpha > 0$ or can be solved optimally in polynomial time. By inspection, the only CSPs that fall into the polynomial-time cases (0-valid, 1-valid, and 2-monotone) and are closed under negation are those containing only dictatorships and constant functions.

The fact that standard PCPs actually yield Levin reductions has been explicitly discussed and formalized by Barak and Goldreich [9] in the terminology of PCPs rather than reductions (the function Enc is called "relatively efficient oracle-construction" and the function Dec is called "a proof-of-knowledge property"). They verify that these properties hold for the PCP construction of Babai et al. [7], whereas we need it for PCPs of constant query complexity. While the properties probably hold for most (if not all) existing PCP constructions, the existence of the efficient "decoding" function g requires some verification. We observe that it follows as a black box from the PCPs of Proximity of [12, 27]. There, a prefix of the PCP (the "implicit input oracle") can be taken to be the encoding of a satisfying assignment of the circuit C in an efficiently decodable error-correcting code. If the PCP verifier accepts with higher probability than the soundness error

s, then it is guaranteed that the prefix is close to a valid codeword, which in turn can be decoded to a satisfying assignment. By the correspondence between PCPs and CSPs [6], this yields a CSP (with constraints of constant arity) that is α-hard to approximate under Levin reductions for some constant $\alpha > 0$ (and $\gamma = 1$). The sequence of approximation-preserving reductions from arbitrary CSPs to MAX-CUT [71] can be verified to preserve efficiency of decoding (indeed, the correctness of the reductions is proven by specifying how to encode and decode). Finally, the reductions of [53] from MAX-CUT to any other CSP all involve constant-sized "gadgets" that allow encoding and decoding to be done locally and very efficiently. □

It seems likely that optimized PCP/inapproximability results (like [47]) are also Levin reductions, which would yield fairly large values for α for natural CSPs (e.g. $\alpha = 1/8 - \varepsilon$ if Γ contains all conjunctions of 3-literals, because then $\mathcal{Q}_\Gamma$ contains MAX 3-SAT.) We are particularly interested in optimizing the *length* of the PCP, in order to establish tighter reductions between generating private synthetic data and forging digital signatures. Below we sketch a proof that a particular construction of short PCPs is also a Levin reduction.

Theorem 4.7. *There exists a fixed family of constant-arity CSPs Γ that is $1/2$-hard-to-approximate under nearly-linear Levin reductions.*

Proof sketch. The existence of a family of constant-arity CSPs Γ that is $1/2$-hard-to-approximate under nearly-linear-length Karp reductions was shown by Dinur [25], building on a PCP construction of Ben-Sasson and Sudan [13]. Thus in order to establish the theorem we only need to verify that satisfying assignments can be encoded and decoded in polynomial time. To this end, we will outline the construction and sketch the encoding and decoding procedures. In the outline we will focus only on the properties of the construction relevant to encoding and decoding of assignments. See [13] and [25] for details regarding the length and soundness parameters of the construction.

1. The first step of the construction is a sequence of reductions from circuit satisfiability to a certain "algebraic CSP"; efficient encoding and decoding of assignments is implicit in the analysis of these reductions. See [13, Section 5.2] and [84, Section 4.3] for details of the reductions.

2. Next, [13] gives a polylogarithmic-query PCP for this algebraic CSP with soundness error $< 1/2$. In the language of CSPs (Definition 4.4), this construction gives a family of

$\text{polylog}(d)$-CSPs that are $1/2$-hard-to-approximate. In their PCP construction, the assignment to the instance of the algebraic CSP—a low-degree univariate polynomial over a finite field—is included as a prefix of the PCP. The remainder of the PCP consists of evaluations of related polynomials on various linear subspaces of the field, which can all be computed in polynomial time (in the size of the field, which is the measure of instance size for the algebraic CSP). The soundness analysis of their construction shows implicitly that any PCP accepted with probability greater than $1/2$ contains a polynomial that is close (in Hamming distance) to a valid satisfying assignment to the instance of the algebraic CSP. That is, the prefix of the PCP that should contain a low-degree univariate polynomial satisfying the algebraic CSP will be close to a unique polynomial that is a valid satisfying assignment. This satisfying assignment can be efficiently decoded from the PCP using polynomial time algorithms for decoding Reed-Solomon codes (see e.g. [14]). See [13] for details of the construction. In particular, see Section 3.3.1 for the argument related to decoding.

3. This PCP can be transformed into a different PCP that only makes two queries to its proof, at the cost of increasing the soundness error to be too large ($1 - o(1)$, rather than $1 - \Omega(1)$) for our desired result. This reduction proceeds in two steps:

   (a) First, in the CSP perspective, we add a new variable (over a large alphabet $\Sigma = \{0,1\}^{\text{polylog}(d)}$) for each constraint. The correct proof will consist of a satisfying assignment to the original CSP and, for each new variable, a copy of all the variables relevant to a particular constraint. The new encoding can clearly be produced efficiently since there are only polynomially many constraints, and thus polynomially many new variables that need to be assigned. The analysis in [25] shows that if any assignment (PCP) satisfies a sufficiently large fraction of the constraints of the new CSP, then the prefix of the assignment corresponding to the original CSP must in fact satisfy a large fraction of the constraints of that CSP. See [25, Section 7.2] for details.

   (b) The second is to reduce the alphabet size back down to binary. This can be accomplished using a standard composition with a PCP of Proximity (see e.g. [4]). The new PCP will consist of an encoding of each symbol with a linear-rate, constant distance, error correcting code that admits efficient encoding and decoding, as well as a polynomial-sized PCP of Proximity admitting efficient encoding. The existence of such

a PCP of Proximity is well-known (and discussed in the context of Theorem 4.6). Here we can afford to use a PCP of Proximity with any polynomial length blowup, since it is applied to symbols of length $\mathrm{polylog}(d)$. Dinur [25] establishes the existence of such a PCP of Proximity using a small number of rounds of gap amplification (see Step 4) and it can easily be seen that efficient encoding for each round of gap amplification holds in the case of PCPs of Proximity as well. See [25, Section 3] for details of composition and Section 8 for a construction of PCPs of Proximity.

4. Finally, we decrease the soundness error to a constant $s < 1$. This can be accomplished, as in [25] via a small number of rounds of "gap amplification." We will check that the CSP produced by each round of gap amplification allows for efficient encoding and decoding. Each step of gap amplification consists of three phases

   (a) A preprocessing phase that ensures the CSP instance has certain useful properties. This phase consists of several simple transformations on the CSP and its assignment and it is implicit in the analysis of these transformations that assignments can be encoded and decoded efficiently. See [25, Section 3.1] for details.

   (b) A "powering" step that creates a new CSP over a larger (but still constant-sized) alphabet. In the correct proof, we replace each variable of the original CSP with a new variable that contains an assignment to every original variable that is "nearby" in an appropriate sense. Thus to encode we simply have to copy the assignment to each variable a small (constant) number of times. The analysis of [25] shows that if an assignment to this new CSP satisfies a large fraction of its constraints, then we can efficiently compute an assignment to the original CSP that satisfies a large fraction of its constraints. Specifically, we assign each variable in the original CSP to be a "plurality vote" of the variables of the new CSP, and this plurality vote, which can be computed efficiently. See [25, Section 5] for details.

   (c) Finally, we want to produce a CSP over a binary alphabet. As in Step 3b, we can achieve this via composition with a PCP of Proximity. However here the alphabet we begin with is constant-sized, so we can afford to use an inefficient PCP of Proximity, and encoding and decoding can be achieved via brute-force enumeration of code words. See [25], Section 6 for details.

□

For some of our results we will need CSPs that are very hard to approximate (under possibly inefficient reductions), which we can obtain by "sequential repetition" of constant-error PCPs.

Theorem 4.8. *There is a constant C such that for every $\varepsilon = \varepsilon(d) > 0$, the constraint family $\Gamma = (\Gamma_d)_{d\in\mathbb{N}}$ of $k(d)$-clause 3-CNF formulas is $(1-\varepsilon(d))$-hard-to-approximate under inefficient Levin reductions, for $k(d) = C\log(1/\varepsilon(d))$.*

Proof sketch. As in the proof of Theorem 4.6, disjunctions of 3 literals are $(1-\delta, 1)$-hard-to-approximate under Levin reductions for some constant $\delta > 0$. By taking $\ell(d) = \log_\delta(\varepsilon(d))$ sequential repetitions of this PCP, we get a PCP with completeness 1 and soundness $\varepsilon(d)$ whose constraints are 3-CNF formulas with $\ell(d) = \log_\delta(1/\varepsilon(d))$ clauses. Note that the resulting CSP will have arity at most $k(d) = 3\ell(d)$.

We have to check that this resulting PCP preserves the properties of inefficient Levin reductions. The encoder for the ℓ-fold sequential repetition is unchanged. If the initial reduction is $R(C) = \varphi_C = \{\varphi_1, \ldots, \varphi_m\}$ (a set of 3-literal disjunctions), then the reduction $R^\ell(C)$ for the ℓ-fold sequential repetition will produce m^ℓ, ℓ-clause 3-CNF formulae by taking every subcollection of ℓ clauses in φ_C. Specifically, for every $i_1, i_2, \ldots, i_\ell \in [m]$, $R^k(C)$ will contain a ℓ-clause 3-CNF formula $\varphi_{i_1} \wedge \varphi_{i_2} \wedge \cdots \wedge \varphi_{i_\ell}$.

The decoder also remains unchanged. If the value of an assignment π is at least δ^ℓ with respect to $R^\ell(C)$ then it must have value at least δ with respect to $R(C)$ and thus $Dec(\pi, C)$ will return a satisfying assignment to C, that is $C(Dec(\pi, C)) = 1$.

Notice that when $k = k(d) = \omega(1)$, the reduction will produce $m^{\omega(1)}$ clauses and be inefficient. Thus we will have an inefficient Levin reduction if we want to obtain $\varepsilon(d) = o(1)$ from this construction. □

4.3 Hard-to-Sanitize Distributions from Hard CSPs

In this section we prove that to efficiently produce a synthetic database that is accurate for the constraints of a CSP that is hard-to-approximate under Levin reductions, we must pay constant error in the worst case. Following [32], we start with a digital signature scheme, and a database of valid message-signature pairs. There is a verifying circuit C_{vk} and valid message-signature pairs

are satisfying assignments to that circuit. Now we encode each row of database using the function Enc, described in Definition 4.5, that maps satisfying assignments to C_{vk} to assignments of the CSP instance $\varphi_{C_{vk}} = R(C_{vk})$ with value at least γ. Then, any assignment to the CSP instance that satisfies a $\gamma - \alpha$ fraction of clauses can be decoded to a valid message-signature pair. The database of encoded message-signature pairs is what we will use as our hard-to-sanitize distribution.

4.3.1 Hardness of Sanitizing

Differential privacy is a very strong notion of privacy, so it is common to look for hardness results that rule out weaker notions of privacy. These hardness results show that every sanitizer must be "blatantly non-private" in some sense. For these results, our notion of blatant non-privacy roughly states that there exists an efficient adversary who can find a row of the original database using only the output from any efficient sanitizer. Such definitions are also referred to as "row non-privacy." We define hardness-of-sanitization with respect to a particular concept class, and want to exhibit a distribution on databases for which it would be infeasible for any efficient sanitizer to give accurate output without revealing a row of the database. Specifically, following [32], we define the following notions

Definition 4.9 (Database Distribution Ensemble)**.** Let $\mathcal{D} = \mathcal{D}_d$ be an ensemble of distributions on d-column databases with $n+1$ rows $D \in (\{0,1\}^d)^{n+1}$. Let $(D, D', i) \leftarrow_{\mathrm{R}} \widetilde{\mathcal{D}}$ denote the experiment in which we choose $D_0 \leftarrow_{\mathrm{R}} \mathcal{D}$ and $i \in [n]$ uniformly at random, and set D to be the first n rows of D_0 and D' to be D with the i-th row replaced by the $(n+1)$-st row of D_0.

Definition 4.10 (Hard-to-sanitize Distribution)**.** Let $\mathcal{Q}$ be a concept class, $\alpha : \mathbb{N} \to [0,1]$ and $T_{\mathsf{San}} : \mathbb{N} \to \mathbb{N}$ be functions, and $\mathcal{D} = \mathcal{D}_d$ be a database distribution ensemble.

The distribution $\mathcal{D}$ is $(\alpha, T_{\mathsf{San}}, \mathcal{Q})$-*hard-to-sanitize* if there exists an efficient adversary $\mathcal{A}$ such that for any alleged sanitizer $\mathcal{M}$ running in time at most $T_{\mathsf{San}}(d)$ the following two conditions hold:

1. Whenever $\mathcal{M}(D)$ is $\alpha(d)$-accurate, then $\mathcal{A}(\mathcal{M}(D))$ outputs a row of D:

$$\Pr_{\substack{(D,D',i)\leftarrow_{\mathrm{R}}\widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ and } \mathcal{A}'s \text{ coins}}} [(\mathcal{M}(D) \text{ is } \alpha(d)\text{-accurate for } \mathcal{Q}) \wedge (\mathcal{A}(\mathcal{M}(D)) \cap D = \emptyset)] \leq \mathsf{negl}(d).$$

2. For every efficient sanitizer $\mathcal{M}$, $\mathcal{A}$ cannot extract $x^{(i)}$ from the database D':

$$\Pr_{\substack{(D,D',i)\leftarrow_{\mathrm{R}}\widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ and } \mathcal{A}'s \text{ coins}}} \left[\mathcal{A}(\mathcal{M}(D')) = x^{(i)}\right] \leq \mathsf{negl}(d)$$

where $x^{(i)}$ is the i-th row of D.

In [32], it was shown that every distribution that is $(\alpha, T_{\mathsf{San}}, \mathcal{Q})$-hard-to-sanitize in the sense of Definition 4.10, is also hard to sanitize while achieving even weak differential privacy

Claim 4.11. *[32] If a distribution ensemble $\mathcal{D} = \mathcal{D}_d$ on $n(d)$-row databases is $(\alpha, T_{\mathsf{San}}, \mathcal{Q})$-hard-to-sanitize, then for every constant $a > 0$ and every $\beta = \beta(d) \leq 1 - 1/\mathrm{poly}(d)$, no $T_{\mathsf{San}}(d)$-time one-shot sanitizer that is (α, β)-accurate with for $\mathcal{Q}$ can achieve $(a\log(n), (1-8\beta)/2n^{1+a})$-differential privacy.*

In particular, for all constants $\epsilon, \beta > 0$, no $T_{\mathsf{San}}(d)$-time one-shot sanitizer can be simultaneously (α, β)-accurate and $(\epsilon, \mathsf{negl}(n))$-differentially private.

We could use a weaker definition of hard-to-sanitize distributions, which would still suffice to rule out differential privacy, and only require that for every efficient $\mathcal{M}$, there exists an adversary $\mathcal{A}_{\mathcal{M}}$ that almost always extracts a row of D from every α-accurate output of $\mathcal{M}(D)$. In our definition we require that there exists a fixed adversary $\mathcal{A}$ that almost always extracts a row of D from every α-accurate output of any efficient $\mathcal{M}$. Reversing the quantifiers in this fashion only makes our negative results stronger.

We are concerned with sanitizers that output synthetic databases, so we will relax Definition 4.10 by restricting the quantification over sanitizers to only those sanitizers that output synthetic data.

Definition 4.12 (Hard-to-sanitize Distribution as Synthetic Data)**.** A database distribution ensemble $\mathcal{D}$ is $(\alpha, T_{\mathsf{San}}, \mathcal{Q})$-*hard-to-sanitize as synthetic data* if the conditions of Definition 4.10 hold for every sanitizer $\mathcal{M}$ that outputs a synthetic database.

The definition of hard-to-sanitize databases can be specialized in a similar way to other output representations besides synthetic data (e.g. medians of synthetic databases).

4.3.2 Super-Secure Digital Signature Schemes

Before proving our main result, we formally define a *super-secure digital signature scheme*. These digital signature schemes have the property that it is infeasible under chosen-message attack to find a new message-signature pair that is different from all those obtained during the attack, even a new signature for an old message. First we formally define digital signature schemes

Definition 4.13 (Digital signature scheme)**.** For a functions $L_{\text{vk}}, T_{\text{Ver}} : \mathbb{N} \to \mathbb{N}$. A $(L_{\text{vk}}, T_{\text{Ver}})$-*digital signature scheme* is a tuple of three polynomial time algorithms $\Pi = (Gen, Sign, Ver)$ such that

1. Gen takes as input the security parameter 1^κ and outputs a key pair $(sk, vk) \leftarrow_{\text{R}} Gen(1^\kappa)$ such that $vk \in \{0,1\}^{L_{\text{vk}}(\kappa)}$ for a polynomial $L_{\text{vk}}(\kappa)$.

2. $Sign$ takes sk and a message $m \in \{0,1\}^*$ as input and outputs $\sigma \leftarrow_{\text{R}} Sign_{sk}(m)$ such that $\sigma \in \{0,1\}^*$.

3. Ver takes vk and pair (m, σ) and deterministically outputs a bit $b \in \{0,1\}$, such that for every (sk, vk) in the range of Gen, and every message m, we have $Ver_{vk}(m, Sign_{sk}(m)) = 1$. Moreover, when $m \in \{0,1\}^\kappa$, and m is signed under (sk, vk) generated by $Gen(1^\kappa)$, then Ver_{vk} can be computed by a circuit of size $T_{\text{Ver}}(\kappa)$ (for a polynomial T_{Ver}).

We define the security of a digital signature scheme with respect to the following game.

Definition 4.14 (Weak forgery game)**.** For any signature scheme $\Pi = (Gen, Sign, Ver)$ and probabilistic polynomial time adversary $\mathcal{F}$, $\texttt{WeakForge}(\mathcal{F}, \Pi, \kappa, Q_{\text{For}})$ is the following probabilistic experiment.

1. $(sk, vk) \leftarrow_{\text{R}} Gen(1^\kappa)$.

2. $\mathcal{F}$ is given vk and oracle access to $Sign_{sk}$. The adversary adaptively queries $Sign_{sk}$ on a set of at most Q_{For} messages $M \subset \{0,1\}^*$, receives a set of message-signature pairs $A \subset \{0,1\}^*$, and outputs (m^*, σ^*).

3. The output of the game is 1 if and only if (1) $Ver_{vk}(m^*, \sigma^*) = 1$, and (2) $(m^*, \sigma^*) \notin A$.

The weak forgery game is easier for the adversary to win than the standard forgery game because the final condition requires that the signature output by $\mathcal{F}$ be different from all pairs $(m, \sigma) \in A$, but allows for the possibility that $m^* \in M$. In the standard definition, the final condition would be replaced by $m^* \notin M$. Thus the adversary has more possible outputs that would result in a "win" under this definition than under the standard definition.

Definition 4.15 (Super-secure digital signature scheme)**.** For functions $T_{\mathrm{For}}, Q_{\mathrm{For}} : \mathbb{N} \to \mathbb{N}$. A $(L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-digital signature scheme $\Pi = (Gen, Sign, Ver)$ is a $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-*super-secure digital signature scheme (under adaptive chosen message attack)* if for every T_{For}-time adversary, $\mathcal{F}$,

$$\Pr[\mathtt{WeakForge}(\mathcal{F}, \Pi, \kappa, Q_{\mathrm{For}}(\kappa)) = 1] \leq \mathsf{negl}(\kappa).$$

Although the above definition is stronger than the usual definition of existentially unforgeable digital signatures, in [40] it is shown how to modify known constructions [66, 75] to obtain a super-secure digital signature scheme from any one-way function.

In our terminology, the existence of one-way functions implies the existence of a digital signature scheme that is a $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-super-secure digital signature scheme for $L_{\mathrm{vk}}, T_{\mathrm{Ver}} = \mathrm{poly}(\kappa)$ and every polynomial $T_{\mathrm{For}}, Q_{\mathrm{For}}$. Under stronger hardness assumptions, super-secure digital signature schemes withe even better parameters exist. In particular, under certain hardness assumptions in ideal lattices, there exists a digital signature scheme for $T_{\mathrm{For}} = 2^{\kappa^{1-o(1)}}$, $L_{\mathrm{vk}} = \kappa^{1+o(1)}$, $T_{\mathrm{Ver}} = \kappa^{1+o(1)}$ and every polynomial Q_{For} [62]. For our results, we need a super-secure digital signature scheme, and we do not know if the scheme of [62], or any other, satisfies this additional property with the same parameters.

4.3.3 A Family of Hard-to-Sanitize Distributions

We are now ready to construct a general form of database distribution ensemble, which we can instantiate with various CSPs and signature schemes to prove our hardness results.

Let $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ be a family of nice $q(d)$-CSPs (Definition 4.3) and let $\mathcal{Q}_\Gamma = \cup_{d=1}^{\infty} \mathcal{Q}_\Gamma^{(d)}$ be the class of constraints of Γ (Definition 4.2). Let $\Pi = (Gen, Sign, Ver)$ be a $(L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-digital signature scheme and let C_{vk} be a circuit computing Ver_{vk}. Let $n : \mathbb{N} \to \mathbb{N}$ be a function.

We define the database distribution ensemble $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$ for any function $n : \mathbb{N} \to \mathbb{N}$. A sample from $\mathcal{D}_d$ consists of $n(d) + 1$ random message-signature pairs encoded as PCP witnesses with respect to the signature-verification algorithm. Each row will also contain an encoding of the verification key for the signature scheme using the non-constant constraint $\varphi^* : \{0,1\}^{q(d)} \to \{0,1\}$ in Γ_d and the assignments $u_0^*, u_1^* \in \{0,1\}^{q(d)}$ such that $\varphi^*(u_0^*) = 0$ and $\varphi^*(u_1^*) = 1$, as described in the definition of nice CSPs (Definition 4.3).

Recall that $s_1 \| s_2$ denotes the concatenation of the strings s_1 and s_2. Before moving on to

Let $n = n(d)$. Let $\varphi^* : \{0,1\}^{q(d)} \to \{0,1\}$ be a non-constant constraint in Γ_d and $u_0^*, u_1^* \in \{0,1\}^{q(d)}$ be such that $\varphi^*(u_0^*) = 0$ and $\varphi^*(u_1^*) = 1$. Let $\kappa = \kappa(d)$ be the largest integer such that $L_{\mathrm{PCP}}(T_{\mathrm{Ver}}(\kappa)) \leq d/2$ and $L_{\mathrm{vk}}(\kappa)q(d) \leq d/2$.

Let $(sk, vk) \leftarrow_{\mathrm{R}} Gen(1^\kappa)$, let $vk = vk_1 vk_2 \ldots vk_\ell$, where $\ell = L_{\mathrm{vk}}(\kappa)$

Let $(m_1, \ldots, m_{n+1}) \leftarrow_{\mathrm{R}} (\{0,1\}^\kappa)^{n+1}$

For: $i = 1$ to $n+1$

Let $y_i := Enc(m_i \| Sign_{sk}(m_i), C_{vk})$, be a PCP encoding of m_i and its signature, padded with zeros to be of length exactly $d/2$

Let $z_i := u^*_{vk_1} \| u^*_{vk_2} \| \ldots \| u^*_{vk_\ell}$, be an encoding of vk, padded with zeros to length $d/2$

Let $x_i := y_i \| z_i$ be the concatenation of these two strings

Return: $D_0 := (x^{(1)}, \ldots, x^{(n+1)})$

Figure 8: Database Distribution Ensemble $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$:

instantiating $\mathcal{D}$ and proving our hardness results, we make some observations about the construction. First, observe that the construction is well defined. That is, the length of y_i before padding is exactly $L_{\mathrm{PCP}}(T_{\mathrm{Ver}}(\kappa))$, and the length of z_i before padding is exactly $L_{\mathrm{vk}}(\kappa)q(d)$ and κ was chosen so that these quantities are both at most $d/2$. Also, note that $\kappa(d) = d^{\Omega(1)}$. This statement holds because Γ is nice (Definition 4.3), so $q(d) = d^{1-\Omega(1)}$, and because $L_{\mathrm{PCP}}, T_{\mathrm{Ver}}$, and L_{vk} are all bounded by some polynomial in their input length. Finally, note that our distribution over d-column databases contains PCPs of length $L = L_{\mathrm{PCP}}(T_{\mathrm{Ver}}(\kappa)) \leq d/2$, thus $R(C_{vk}) \subseteq \mathcal{Q}_\Gamma^{(d/2)} \subseteq \mathcal{Q}_\Gamma^{(d)}$ (by Definition 4.2).

4.3.4 Main Hardness Result

We are now ready to state and prove our main hardness result.

Theorem 4.16. *Let $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ be a family of nice (Definition 4.3) $q(d)$-CSPs such that $\Gamma_d \cup \neg\Gamma_d$ is α-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup L_{PCP} for $\alpha = \alpha(d) \in (0, 1/2)$. Assume the existence of a $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-digital signature scheme, Π. Let $\kappa : \mathbb{N} \to \mathbb{N}$ be as defined in the construction of $\mathcal{D}$ and let $\kappa = \kappa(d)$. Let $T_{\mathsf{San}}, n : \mathbb{N} \to \mathbb{N}$ be*

any functions such that

$$n(d) \cdot d^a + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$$

for any $a > 0$. *and*

$$n(d) \leq Q_{\mathrm{For}}(\kappa(d))$$

for every $d \in \mathbb{N}$. *Then the distribution ensemble* $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$ *on* $n(d)$*-row databases is* $(\alpha, T_{\mathsf{San}}(d), \mathcal{Q}_\Gamma^{(d)})$*-hard-to-sanitize as synthetic data.*

Proof. Let $\Pi = (Gen, Sign, Ver)$ be the assumed $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-super-secure digital signature scheme and let C_{vk} be a circuit computing Ver_{vk}. Let Γ be the assumed family of nice $q(d)$-CSPs that is α-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup L_{PCP}. Let R, Enc, Dec be the functions corresponding to the Levin reduction to Γ and $\gamma = \gamma(d) \in (\alpha, 1]$ be the parameter from Definition 4.5.

Every valid pair $(m, Sign_{sk}(m))$ is a satisfying assignment of the circuit C_{vk}, hence every row of D_0 constructed by $\mathcal{D}$ will satisfy at least a γ fraction of the clauses of the formula $\varphi_{C_{vk}} = R(C_{vk}) \subseteq \mathcal{Q}_\Gamma^{(d)}$. Additionally, for every bit of the verification key, there is a block of $q = q(d)$ bits in each row that contains either a satisfying assignment or a non-satisfying assignment of φ^*, depending on whether that bit of the key is 1 or 0. Specifically, define the predicates $\varphi_j^*(x) = \varphi^*(x_{d/2+(j-1)q+1}, x_{d/2+(j-1)q+2}, \ldots, x_{d/2+jq})$ for $j = 1, 2, \ldots, L_{\mathrm{vk}}(\kappa)$. Then, by construction, $\varphi_j^*(D_0) = vk_j$, the j-th bit of the verification key. Note that $\varphi_j^* \in \mathcal{Q}_\Gamma^{(d)}$ for $j = 1, 2, \ldots, \ell$, by our construction of $\mathcal{Q}_\Gamma^{(d)}$ (Definition 4.2).

We now prove the following two lemmas that will establish $\mathcal{D}$ is hard-to-sanitize:

Lemma 4.17. *There exists a polynomial-time adversary* $\mathcal{A}$ *such that for every* T_{San}*-time sanitizer* $\mathcal{M}$*,*

$$\Pr_{\substack{(D,D',i)\leftarrow_R \widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ and } \mathcal{A}'s \text{ coins}}} \left[(\mathcal{M}(D) \textit{ is } \alpha\textit{-accurate for } \mathcal{Q}_\Gamma^{(d)}) \wedge (\mathcal{A}(\mathcal{M}(D)) \cap D = \emptyset) \right] \leq \mathsf{negl}(d) \qquad (4.1)$$

Proof. Our privacy adversary tries to find a row of the original database by trying to PCP-decode each row of the "sanitized" database and then re-encoding it. In order to do so, the adversary needs to know the verification key used in the construction of the database, which it can discover from the answers to the queries φ_j^*, defined above. Formally, we define the privacy adversary by means of a subroutine that tries to learn the verification key and then PCP-decode each row of the input database:

Subroutine $\mathcal{K}(\widehat{D})$:

Let d be the dimension of rows in $\widehat{D}$, let κ be as in the construction of $\mathcal{D}$, and $\ell = L_{\mathrm{vk}}(\kappa)$.

For: $j = 1$ to ℓ

$\widehat{vk}_j = \left[\varphi_j^*(\widehat{D}) \text{ rounded to } \{0,1\}\right]$

Return: $\widehat{vk}_1 \| \widehat{vk}_2 \| \ldots \| \widehat{vk}_\ell$

Subroutine $\mathcal{A}_0(\widehat{D})$:

Let $\widehat{n}$ be the number of rows in $\widehat{D}$, $\widehat{vk} = \mathcal{K}(\widehat{D})$

For: $i = 1$ to $\widehat{n}$

If: $C_{\widehat{vk}}(Dec(\widehat{x}^{(i)}, C_{\widehat{vk}})) = 1$

Return: $Dec(\widehat{x}^{(i)}, C_{\widehat{vk}})$

Return: $\perp$

Privacy Adversary $\mathcal{A}(\widehat{D})$:

Let $\widehat{vk} = \mathcal{K}(\widehat{D})$.

Return: $Enc(\mathcal{A}_0(\widehat{D}), C_{\widehat{vk}})$

Figure 9: An adversary for private synthetic data.

Let $\mathcal{M}$ be a T_{San}-time sanitizer, we will show that Inequality (4.1) holds.

Claim 4.18. *If $\widehat{D} = \mathcal{M}(D)$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$, then $\mathcal{A}_0(\widehat{D})$ will output a pair (m, σ) s.t. $C_{vk}(m, \sigma) = 1$.*

Proof. First we argue that if $\widehat{D}$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$ for $\alpha < 1/2$, then $\mathcal{K}(\widehat{D}) = vk$, where vk is the verification key used in the construction of D_0. By construction, $\varphi_j^*(D) = vk_j$. If $vk_j = 0$ and $\widehat{D}$ is α-accurate for D then $\varphi_j^*(\widehat{D}) \leq \alpha < 1/2$, and $\widehat{vk}_j = vk_j$. Similarly, if $vk_j = 1$ then $\varphi_j^*(\widehat{D}) \geq 1 - \alpha > 1/2$, and $\widehat{vk}_j = vk_j$. Thus, for the rest of the proof we will be justified in substituting vk for $\widehat{vk}$.

Next we show that if $\widehat{D}$ is α-accurate, then $\mathcal{A}_0(\widehat{D}) \neq \perp$. It is sufficient to show there exists $\widehat{x}^{(i)} \in \widehat{D}$ such that $\mathrm{val}(\varphi_{C_{vk}}, x^{(i)}) \geq \gamma - \alpha$, which implies $C_{vk}(Dec(\widehat{x}^{(i)}, C_{vk})) = 1$.

Since every $(m_i, Sign_{sk}(m_i))$ pair is a satisfying assignment to C_{vk}, the definition of Enc (Definition 4.5) implies that each row $x^{(i)}$ of D has $\mathrm{val}(\varphi_{C_{vk}}, x^{(i)}) \geq \gamma$. Thus if $\varphi_{C_{vk}} = \{\varphi_1, \ldots, \varphi_m\}$,

then

$$\frac{1}{m}\sum_{j=1}^{m}\varphi_j(D) = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}\varphi_j(x^{(i)})\right) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{val}(\varphi_{C_{vk}}, x^{(i)}) \geq \gamma.$$

Since $\widehat{D}$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$, and for every constraint φ_j, either $\varphi_j \in \Gamma$ or $\neg\varphi_j \in \Gamma$, then for every constraint $\varphi_j \in \varphi_{C_{vk}}$, we have $\varphi_j(\widehat{D}) \geq \varphi_j(D) - \alpha$. Thus

$$\frac{1}{\widehat{n}}\sum_{i=1}^{\widehat{n}}\mathrm{val}(\varphi_{C_{vk}}, \widehat{x}^{(i)}) = \frac{1}{m}\sum_{j=1}^{m}\varphi_j(\widehat{D}) \geq \frac{1}{m}\sum_{j=1}^{m}\varphi_j(D) - \alpha \geq \gamma - \alpha.$$

So for at least one row $\widehat{x}^{()} \in \widehat{D}$ it must be the case that $\mathrm{val}(\varphi_{C_{vk}}, \widehat{x}^{()}) \geq \gamma - \alpha$. The definition of Dec (Definition 4.5) implies $C_{vk}(Dec(\widehat{x}^{()}, C_{vk})) = 1$. □

Now notice that if $\mathcal{A}_0(\mathcal{M}(D))$ outputs a valid message-signature pair but $\mathcal{A}(\mathcal{M}(D)) \cap D = \emptyset$, then this means $\mathcal{A}_0(\mathcal{M}(D))$ is forging a new signature not among those used to generate D. Formally, we construct a signature forger as follows:

Use the oracle $Sign_{sk}$ to generate an n-row database D just as in the definition of $\mathcal{D}_d$ (consisting of PCP encodings of valid message-signature pairs and an encoding of vk).
Let $\widehat{D} := \mathcal{M}(D)$
Return: $\widehat{x}^{(*)} := \mathcal{A}_0(\widehat{D})$

Figure 10: Forger $\mathcal{F}(vk)$ with oracle access to $Sign_{sk}$.

First we analyze the running time of $\mathcal{F}$. In order to construct D, the forger must PCP-encode the signatures, which requires time $n(d) \cdot \mathrm{poly}(T_{\mathrm{Ver}}(\kappa)) = n(d) \cdot \mathrm{poly}(d)$. Running $\mathcal{M}$ requires time $T_{\mathsf{San}}(d)$ by assumption. Finally, the time to run $\mathcal{A}_0(\widehat{D})$ to decode a message-signature pair is $n(d) \cdot \mathrm{poly}(d)$, since $\mathcal{A}_0$ has to run the PCP decoder on each row of $\widehat{D}$. Put together, the running time of $\mathcal{F}$ is $n(d) \cdot \mathrm{poly}(d) + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$, where the inequality is an assumption of the theorem. Next, we note that the forger makes at most $n(d) \leq o(Q_{\mathrm{For}}(\kappa(d)))$ queries to $Sign_{sk}$, and this inequality is also an assumption of the theorem.

Now observe that running $\mathcal{F}$ in the experiment $\mathtt{WeakForge}(\mathcal{F}, \Pi, \kappa, Q_{\mathrm{For}}(\kappa))$ is equivalent to running $\mathcal{A}$ in the experiment of Inequality (4.1), except that $\mathcal{F}$ does not re-encode the output of $\mathcal{A}_0(\mathcal{M}(D))$. By the super-security of the signature scheme, if the $\widehat{x}^{(*)}$ output by $\mathcal{F}$ is a valid

message-signature pair (as holds if $\mathcal{M}(D)$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$, by Claim 4.18), then it must be one of the message-signature pairs used to construct D (except with probability $\mathsf{negl}(\kappa) = \mathsf{negl}(d^{\Omega(1)}) = \mathsf{negl}(d)$). This implies that $\mathcal{A}(\mathcal{M}(D)) = Enc(\widehat{x}^{(*)}, C_{vk}) \in D$ (except with negligible probability). Thus, we have

$$\Pr_{\substack{(D,D',i)\leftarrow_R \widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ coins}}}[\mathcal{M}(D) \text{ is } \alpha\text{-accurate for } \mathcal{Q}_\Gamma^{(d)} \Rightarrow \mathcal{A}(\mathcal{M}(D)) \in D] \geq 1 - \mathsf{negl}(d),$$

which is equivalent to the statement of the lemma. □

Lemma 4.19.

$$\Pr_{\substack{(D,D',i)\leftarrow_R \widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ and } \mathcal{A}'s \text{ coins}}} \left[\mathcal{A}(\mathcal{M}(D')) = x^{(i)}\right] \leq \mathsf{negl}(d)$$

Proof. Since the messages m_i used in D_0 are drawn independently, D' contains no information about the message m_i, thus no adversary can, on input $\mathcal{M}(D')$ output the target row $x^{(i)}$ except with probability $2^{-\kappa} = \mathsf{negl}(d)$. □

These two claims suffice to establish that $\mathcal{D}$ is $(\alpha, \mathcal{Q}_\Gamma)$-hard-to-sanitize as synthetic data. □

Theorem 4.1 in the introduction follows by combining Theorems 4.6 and 4.16.

If we assume the existence of an efficient digital signature scheme secure against nearly-exponential-time adversaries then we obtain the following variant of Theorem 4.1.

Corollary 4.20. *Assume the existence of a digital signature scheme that is $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-super-secure for $T_{\mathrm{For}} = 2^{\kappa^{1-o(1)}}$, $L_{\mathrm{vk}} = \kappa^{1+o(1)}$, $T_{\mathrm{Ver}} = \kappa^{1+o(1)}$ and every polynomial Q_{For}. Then there exists a family Γ of constant-arity CSPs such that for every polynomial n, the distribution ensemble $\mathcal{D} = \mathcal{D}_d(d, \Gamma, \Pi)$ on $n(d)$-row databases is $(1/2, T_{\mathsf{San}}, \mathcal{Q}_\Gamma^{(d)})$-hard-to-sanitize as synthetic data for some $T_{\mathsf{San}}(d) = 2^{d^{1-o(1)}}$.*

Proof. By Theorem 4.7, there exists a fixed family of constant-arity q-CSPs that is $1/2$-hard-to-approximate under Levin reductions with nearly-linear length blowup. That is, $L_{\mathrm{PCP}}(s) = s^{1+o(1)}$. Now in the construction of $\mathcal{D}$, we can choose $\kappa = d^{1-o(1)}$ such that $L_{\mathrm{PCP}}(T_{\mathrm{Ver}}(\kappa)) \leq d/2$ and $L_{\mathrm{vk}} \cdot q \leq d/2$. Thus we have $T_{\mathrm{For}}(\kappa) = 2^{\kappa^{1-o(1)}} = 2^{d^{1-o(1)}}$.

Now we can find some function $T_{\mathsf{San}}(d) = 2^{d^{1-o(1)}}$, such that

$$n(d) \cdot d^a + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$$

for every $a > 0$. Additionally, for every polynomial $n(d)$, we will have

$$n(d) \leq Q_{\mathrm{For}}(\kappa(d)).$$

Thus the assumptions of Theorem 4.16 are satisfied. □

4.4 Relaxed Synthetic Data

The proof of Theorem 4.16 requires that the sanitizer output a synthetic database. In this section we present similar hardness results for sanitizers that produce other forms of output, as long as they still produce a collection of elements from $\{0,1\}^d$, that are interpreted as the data of (possibly "fake") individuals. More specifically, we consider sanitizers that output a database $\widehat{D} \in (\{0,1\}^d)^{\widehat{n}}$ but are interpreted using an evaluation function of the following form: To evaluate predicate $q \in \mathcal{Q}$ on $\widehat{D}$, apply q to each row $\widehat{x}^{(i)}$ of $\widehat{D}$ to get a string of $\widehat{n}$ bits, and then apply a function $f : \{0,1\}^{\widehat{n}} \times \mathcal{Q} \to [0,1]$ to determine the answer. For example, when the sanitizer outputs a synthetic database, we have $f(b_1, \ldots, b_{\widehat{n}}, q) = (1/\widehat{n}) \sum_{i=1}^{\widehat{n}} b_i$, which is just the fraction of rows that get labeled with a 1 by the predicate q (independent of q).

We now give a formal definition of *relaxed synthetic data*

Definition 4.21 (Relaxed Synthetic Data)**.** A sanitizer $\mathcal{M} : (\{0,1\}^d)^n \to (\{0,1\}^d)^{\widehat{n}}$ with evaluator $\mathcal{E}$ outputs *relaxed synthetic data* for a family of predicates $\mathcal{Q}$ if there exists $f : \{0,1\}^{\widehat{n}} \times \mathcal{Q} \to [0,1]$ such that

- For every $q \in \mathcal{Q}$

  $$\mathcal{E}(\widehat{D}, q) = f(q(\widehat{x}^{(1)}), q(\widehat{x}^{(2)}), \ldots, q(\widehat{x}^{(\widehat{n})}), q),$$

  and

- f is monotone[9] in the first $\widehat{n}$ inputs.

This relaxed notion of synthetic data is of interest because many natural approaches to sanitizing yield outputs of this type. In particular, several previous sanitization algorithms [16, 78, 35] produce a *set* of synthetic databases and answer a query by taking a median over the answers given

[9]Given two vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ we say $b \succeq a$ iff $b_i \geq a_i$ for every $i \in [n]$. We say a function $f : \{0,1\}^n \to [0,1]$ is *monotone* if $b \succeq a \implies f(b) \geq f(a)$.

by the individual databases. Sanitizers that use medians of synthetic databases no longer have the advantage that they are "interchangeable" with the original data, but are still be desirable for data releases because they retain the property that a small data structure can give accurate answers to a large number of queries. We view such databases as a single synthetic database but require that f have a special form. Unfortunately, the sanitizers of [78] and [35] run in time exponential in the dimension of the data, d, and the results of the next subsection show this limitation is inherent even for simple concept classes.

Throughout this section we will continue to use $c(\widehat{D})$ to refer to the answer given by $\widehat{D}$ *when interpreted as a synthetic database*.

We now present our hardness results for relaxed synthetic data where the function f takes the median over synthetic database (Section 4.4.1), where f is an arbitrary monotone, symmetric function (Section 4.4.2), or when the family of concepts contains CSPs that are very hard to approximate (Section 4.4.3). Our proofs use the same construction of hard-to-sanitize databases as Theorem 4.16 with a modified analysis and parameters to show that the output must still contain a PCP-decodable row.

4.4.1 Hardness of Sanitizing as Medians

In this section we establish that the distribution $\mathcal{D}$ used in the proof of Theorem 4.16 is hard-to-sanitize as medians of synthetic data, formally defined as:

Definition 4.22 (Medians of Synthetic Data)**.** A sanitizer $\mathcal{M} : (\{0,1\}^d)^n \to (\{0,1\}^d)^{\widehat{n}}$ with evaluator $\mathcal{E}$ outputs *medians of synthetic data* if there is a partition $[\widehat{n}] = S_1 \cup S_2 \cdots \cup S_\ell$ such that

$$\mathcal{E}(\widehat{x}^{(1)}, \ldots, \widehat{x}^{(\widehat{n})}, q) = \text{median}\left\{ \frac{1}{|S_1|}\sum_{i \in S_1} q(\widehat{x}^{(i)}), \frac{1}{|S_2|}\sum_{i \in S_2} q(\widehat{x}^{(i)}), \ldots, \frac{1}{|S_\ell|}\sum_{i \in S_\ell} q(\widehat{x}^{(i)}) \right\}.$$

Note that medians of synthetic data are a special case of relaxed synthetic data. In the following, we rule out efficient sanitizers with medians of synthetic data for CSPs that are hard to approximate within a multiplicative factor larger than 2. By Theorem 4.8, these CSPs include k-clause 3-CNF formulas for some constant k.

Theorem 4.23. *Let $\Gamma = (\Gamma_d)_{d \in \mathbb{N}}$ be a family of nice (Definition 4.3) $q(d)$-CSPs such that $\Gamma_d \cup \neg\Gamma_d$ is $((\alpha+\gamma)/2, \gamma)$-hard-to-approximate under (possibly inefficient) Levin reductions with length*

blowup L_{PCP} for $\alpha = \alpha(d) \in (0, 1/2)$. Assume the existence of a $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-digital signature scheme, Π. Let $\kappa : \mathbb{N} \to \mathbb{N}$ be as defined in the construction of $\mathcal{D}$ and let $\kappa = \kappa(d)$. Let $T_{\mathsf{San}}, n : \mathbb{N} \to \mathbb{N}$ be any functions such that

$$n(d) \cdot d^a + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$$

for any $a > 0$. and

$$n(d) \leq Q_{\mathrm{For}}(\kappa(d))$$

for every $d \in \mathbb{N}$. Then the distribution ensemble $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$ on $n(d)$-row databases is $(\alpha, T_{\mathsf{San}}(d), \mathcal{Q}_\Gamma^{(d)})$-hard-to-sanitize as medians of synthetic data.

Proof. Let Γ be the assumed family of nice $q(d)-$CSPs that is $((\alpha+\gamma)/2, \gamma)$-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup L_{PCP}. Let R, Enc, Dec be the functions corresponding to the Levin reduction to Γ. Let $\mathcal{M}(D) = \widehat{D}$ and let $\left\{\widehat{D}_1, \widehat{D}_2, \ldots, \widehat{D}_\ell\right\}$ be the partition of the rows of $\widehat{D}$ corresponding to $S_1, \ldots, S_\ell$, i.e. $\widehat{D}_i = (\widehat{x}^{(j)})_{j \in S_i}$.

Assuming that $\widehat{D}$ is α-accurate as medians of synthetic data, we will show that there must exist a row $\widehat{x}^{()} \in \widehat{D}$ such that $\mathrm{val}(\varphi_{C_{vk}}, \widehat{x}^{()}) \geq \gamma - (\alpha + \gamma)/2 = (\gamma - \alpha)/2$. To do so, we observe that if $\widehat{D}$ is accurate as medians of synthetic databases, then for each predicate, half of $\widehat{D}$'s synthetic databases must give an answer that is "close to D's answer". Thus one of these synthetic datsbases must be "close" to D for half of the predicates in $\varphi_{C_{vk}}$. By our construction of D, we conclude that each of these predicates is satisfied by many rows of this synthetic database and thus some row satisfies enough of the predicates to decode a message-signature pair.

We need to show that there exists an adversary $\mathcal{A}$ such that for every T_{San}-time sanitizer $\mathcal{M}$,

$$\Pr_{\substack{(D, D', i) \leftarrow_R \widetilde{\mathcal{D}} \\ \mathcal{M}'s \text{ and } \mathcal{A}'s \text{ coins}}} \left[(\mathcal{M}(D) \text{ is } \alpha\text{-accurate for } \mathcal{Q}_\Gamma^{(d)}) \wedge (\mathcal{A}(\mathcal{M}(D)) \cap D = \emptyset) \right] \leq \mathsf{negl}(d) \tag{4.2}$$

To do so, we will use the same subroutine $\mathcal{A}_0(\widehat{D})$ we used for the proof of Lemma 4.17. That is, we consider a subroutine that looks for rows satisfying sufficiently many clauses of $\varphi_{C_{vk}}$ and returns the PCP-decoding of that row. It will suffice to establish the following claim, analogous to Claim 4.18:

Claim 4.24. *If $\widehat{D}$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$ as medians of synthetic data, then $\mathcal{A}_0(\widehat{D})$ outputs a pair (m, σ) s.t. $C_{vk}(m, \sigma) = 1$.*

Proof. As in the proof of Claim 4.18, if $\widehat{D}$ is α-accurate for $\mathcal{Q}_\Gamma^{(d)}$ for $\alpha < 1/2$, then $\mathcal{K}(\widehat{D}) = vk$, the verification key used in the construction of D_0. For the rest of the proof we will be justified in substituting vk for $\widehat{vk}$.

If $\varphi_{C_{vk}} = \{\varphi_1, \ldots, \varphi_m\}$, then $\frac{1}{m}\sum_{j=1}^{m} \varphi_j(D) \geq \gamma$. We say that $\widehat{D}_k$ is *good for* φ_j if $\varphi_j(\widehat{D}_k) \geq \varphi_j(D) - \alpha$. Since the median over $\left\{\widehat{D}_1, \widehat{D}_2, \ldots, \widehat{D}_\ell\right\}$ is α-accurate for every constraint $\varphi_j \in \varphi_{C_{vk}}$ we have

$$\Pr_{k \leftarrow_R [\ell]}\left[\widehat{D}_k \text{ is good for } \varphi_j\right] \geq \frac{1}{2}$$

Then

$$\begin{aligned}
\mathop{\mathbb{E}}_{k \leftarrow_R [\ell]}\left[\frac{1}{|S_k|}\sum_{i \in |S_k|} \mathrm{val}(\varphi_{C_{vk}}, \widehat{x}^{(i)})\right] &= \mathop{\mathbb{E}}_{k \leftarrow_R [\ell]}\left[\frac{1}{m}\sum_{j=1}^{m} \varphi_j(\widehat{D}_k)\right] \\
&= \frac{1}{m}\sum_{j=1}^{m} \mathop{\mathbb{E}}_{k \leftarrow_R [\ell]}\left[\varphi_j(\widehat{D}_k)\right] \\
&\geq \frac{1}{m}\sum_{j=1}^{m}\left(\Pr_{k \leftarrow_R [\ell]}\left[\widehat{D}_k \text{ is good for } \varphi_j\right] \cdot ((\varphi_j(D) - \alpha)\right) \\
&\geq \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{2} \cdot (\varphi_j(D) - \alpha)\right) \\
&\geq \frac{\gamma - \alpha}{2}
\end{aligned}$$

□

So for at least one row $\widehat{x}^{()} \in \widehat{D}$ it must be the case that $\mathrm{val}(\varphi_{C_{vk}}, \widehat{x}^{()}) \geq (\gamma - \alpha)/2$. Since the distribution $\mathcal{D}$ is unchanged, Lemma 4.19 still holds in this setting. Thus we have established that $\mathcal{D}$ is $(\alpha(d), T_{\mathsf{San}}(d), \mathcal{Q}_\Gamma^{(d)})$-hard-to-sanitize as medians of synthetic data. □

4.4.2 Hardness of Sanitizing with Symmetric Evaluation Functions

In this section we establish the hardness of sanitization for relaxed synthetic data where the evaluator function is symmetric.

Definition 4.25 (Symmetric Relaxed Synthetic Data)**.** A sanitizer $\mathcal{M} : (\{0,1\}^d)^n \to (\{0,1\}^d)^{\widehat{n}}$ with evaluator $\mathcal{E}$ outputs *symmetric relaxed synthetic data* if there exists a monotone function

$g : [0,1] \to [0,1]$ such that

$$\mathcal{E}(\widehat{x}^{(1)}, \ldots, \widehat{x}^{(\widehat{n})}, q) = g\left(\frac{1}{\widehat{n}} \sum_{i=1}^{\widehat{n}} q(\widehat{x}^{(i)})\right).$$

Note that symmetric relaxed synthetic data is also a special case of relaxed synthetic data. Our definition of symmetric relaxed synthetic data is actually symmetric in two respects, because we require that g does not depend on the predicate q and also that g only depends on the fraction of rows that satisfy q. Similar to medians of synthetic data,we show that it is intractable to produce a sanitization as symmetric relaxed synthetic data that is accurate when the queries come from a CSP that is hard to approximate.

Theorem 4.26. *Let $\Gamma = (\Gamma_d)_{d\in\mathbb{N}}$ be a family of nice (Definition 4.3) $q(d)$-CSPs that is closed under complement ($\Gamma_d = \neg\Gamma_d$) and is $(\alpha + 1/2)$-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup L_{PCP} for $\alpha = \alpha(d) \in (0, 1/2)$. Assume the existence of a $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$-digital signature scheme, Π. Let $\kappa : \mathbb{N} \to \mathbb{N}$ be as defined in the construction of $\mathcal{D}$ and let $\kappa = \kappa(d)$. Let $T_{\mathsf{San}}, n : \mathbb{N} \to \mathbb{N}$ be any functions such that*

$$n(d) \cdot d^a + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$$

for any $a > 0$. and

$$n(d) \leq Q_{\mathrm{For}}(\kappa(d))$$

for every $d \in \mathbb{N}$. Then the distribution ensemble $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$ on $n(d)$-row databases is $(\alpha, T_{\mathsf{San}}(d), \mathcal{Q}_\Gamma^{(d)})$-hard-to-sanitize as symmetric relaxed synthetic data.

By Theorem 4.8, the family of k-clause CNF formulas, for some constant k, is $(1/2+\alpha)$-hard-to-approximate under Levin reductions for $\alpha > 0$.

Proof. Let Γ be the assumed family of nice $q(d) -$ CSPs that is $(\alpha + 1/2)$-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup L_{PCP}. Let R, Enc, Dec be the functions corresponding to the Levin reudction to Γ and $\gamma = \gamma(d) \in (\alpha, 1]$ be the parameter from Definition 4.5.

We will use the same approach as in the proof ofTheorem 4.16, which is to show that the underlying synthetic database cannot contain a row that satisfies too many clauses of $\varphi_{C_{vk}}$, in

order to show that g must map a small input to a large output and a large input to a small answer, contradicting the monotonicity of g.

Let $\varphi_{C_{vk}} = \{\varphi_1, \ldots, \varphi_m\}$, then $\frac{1}{m}\sum_{j=1}^{m} \varphi_j(D) \geq \gamma$. It must also be that $\frac{1}{m}\sum_{j=1}^{m} \varphi_j(\widehat{D}) \leq \gamma - \alpha - 1/2$. Otherwise there would exist a row $\widehat{x}^{()} \in \widehat{D} = \mathcal{M}(D)$ such that $\mathrm{val}(\varphi_{C_{vk}}) \geq \gamma - \alpha - 1/2$. But if this were the case we could PCP-decode $\widehat{x}^{()}$ as in the proof of Theorem 4.16. Thus

$$\frac{1}{m}\sum_{j=1}^{m}\left(\varphi_j(D) - \varphi_j(\widehat{D})\right) \geq \alpha + 1/2$$

so there must exist $J \in [m]$ s.t. $\varphi_J(D) - \varphi_J(\widehat{D}) \geq \alpha + 1/2$. Since $\varphi_J(\widehat{D}) \geq 0$ we also have $\varphi_J(D) \geq \alpha + 1/2 > 1/2$, and since $\varphi_J(D) \leq 1$ we also have $\varphi_J(\widehat{D}) \leq 1/2 - \alpha < 1/2$.

By monotonicity of g and α-accuracy of $\widehat{D}$ as symmetric relaxed synthetic data we have

$$g(1/2 - \alpha) \geq g(\varphi_J(\widehat{D})) \geq \varphi_J(D) - \alpha \geq \frac{1}{2}.$$

Consider the negation of φ_J. Since $\neg\varphi_J(D) = 1 - \varphi_J(D)$ we can conclude that $\neg\varphi_J(D) \leq 1/2 - \alpha$ and $\neg\varphi_J(\widehat{D}) \geq 1/2 + \alpha$. Thus we have

$$g(1/2 + \alpha) \leq g(\neg\varphi_J(\widehat{D})) \leq \neg\varphi_J(D) + \alpha \leq \frac{1}{2}.$$

But $g(1/2 - \alpha) \geq 1/2 \geq g(1/2 + \alpha)$ and $\alpha > 0$ contradicts the monotonicity of g. □

4.4.3 Hardness of Sanitizing Very Hard CSPs with Relaxed Synthetic Data

In this section we show that no efficient sanitizer can produce accurate relaxed synthetic data for a sequence of CSPs that is $(1 - \mathsf{negl}(d))$-hard-to-approximate under inefficient Levin reductions. By Theorem 4.8, these CSPs include 3-CNF formulas of $\omega(\log d)$ clauses.

Intuitively, an efficient sanitizer must produce a synthetic database of $\widehat{n}(d) = \mathrm{poly}(d)$ rows, and thus as d grows, an efficient sanitizer cannot produce a synthetic database that contains a row satisfying a non-negligible fraction of clauses from a particular CSP instance (the signature-verification CSP from our earlier results). Thus using evaluators of the type in Definition 4.21 there can only be one answer to most queries, and thus we cannot get an accurate sanitizer.

Theorem 4.27. *Let $\Gamma = (\Gamma_d)_{d\in\mathbb{N}}$ be a family of nice (Definition 4.3) $q(d)$-CSPs such that $\Gamma_d \cup \neg\Gamma_d$ is $(1-\varepsilon, 1)$-hard-to-approximate under (possibly inefficient) Levin reductions with length blowup*

L_{PCP} *for a negligible function* $\varepsilon = \varepsilon(d)$*. Assume the existence of a* $(T_{\mathrm{For}}, Q_{\mathrm{For}}, L_{\mathrm{vk}}, T_{\mathrm{Ver}})$*-digital signature scheme,* Π*. Let* $\kappa : \mathbb{N} \to \mathbb{N}$ *be as defined in the construction of* $\mathcal{D}$ *and let* $\kappa = \kappa(d)$*. Let* $T_{\mathsf{San}}, n : \mathbb{N} \to \mathbb{N}$ *be any functions such that*

$$n(d) \cdot d^a + T_{\mathsf{San}}(d) = o(T_{\mathrm{For}}(\kappa(d)))$$

for any $a > 0$*. and*

$$n(d) \leq Q_{\mathrm{For}}(\kappa(d))$$

for every $d \in \mathbb{N}$*. Then the distribution ensemble* $\mathcal{D} = \mathcal{D}_d(n, \Gamma, \Pi)$ *on* $n(d)$*-row databases is* $(1/3, T_{\mathsf{San}}(d), \mathcal{Q}_{\Gamma}^{(d)})$*-hard-to-sanitize as relaxed synthetic data.*

Proof. Let Γ be the assumed family of nice $q(d)$ − CSPs that is $(1 - \varepsilon(d), 1, L_{\mathrm{PCP}})$-hard-to-approximate under (possibly inefficient) Levin reductions. Let R, Enc, Dec be the functions corresponding to the Levin reudction to Γ. Let $D \leftarrow_{\mathrm{R}} \mathcal{D}_d$, and $\mathcal{M}(D) = \widehat{D}$.

Let $\varphi_{C_{vk}} = \{\varphi_1, \ldots, \varphi_m\}$. By the construction of $\mathcal{D}_d$ we have

$$\varphi_j(D) = 1$$

for every $j \in [m]$. As in the proof of Theorem 4.26 it must be that

$$\frac{1}{m} \sum_{j=1}^{m} \varphi_j(\widehat{D}) \leq \varepsilon(d).$$

Otherwise there would exist a row $\widehat{x}^{()} \in \widehat{D} = \mathcal{M}(D)$ such that $\mathrm{val}(\varphi_{C_{vk}}) \geq \varepsilon(d)$. But if this were the case we could PCP-decode $\widehat{x}^{()}$ as in the proof of Theorem 4.16.

Since $\mathbb{E}_{j \leftarrow_{\mathrm{R}}[m]}[\varphi_j(\widehat{D})] \leq \varepsilon(d)$, there must exist a subset $J \subseteq [m]$ of size $|J| \geq 2m/3$ such that for all $j \in J$, $\varphi_j(\widehat{D}) \leq 3\varepsilon(d) \leq \mathsf{negl}(d)$.

Since $\widehat{D} \in (\{0,1\}^d)^{\widehat{n}}$ for $\widehat{n} = \widehat{n}(d) = \mathrm{poly}(d)$ (by the efficiency of $\mathcal{M}$),

$$\frac{1}{\widehat{n}} \sum_{i=1}^{\widehat{n}} \varphi_j(\widehat{x}^{(i)}) = \varphi_j(\widehat{D}) \in \{0, 1/\widehat{n}(d), 2/\widehat{n}(d), \ldots, 1\}$$

and thus $\varphi_j(\widehat{D}) \leq \mathsf{negl}(d)$ implies $\varphi_j(\widehat{D}) = 0$ for large n.

Let $\mathcal{E}(\widehat{D}, q) = f(q(\widehat{x}^{(1)}), \ldots, q(\widehat{x}^{(\widehat{n})}), q)$. If we assume $\widehat{D}$ is $1/3$-accurate as relaxed synthetic data then $f(0^{\widehat{n}}, \varphi_j) \geq 2/3$ for every $j \in J$.

Now consider the execution of $\mathcal{M}$ on the database $D' = (0^d)^n$. With probability $1 - \mathsf{negl}(d)$, $Dec(0^d, C_{vk})$ is not a valid message-signature pair, thus by Definition 4.5 Part 3, we have

$$\varphi_{C_{vk}}(D') = \varphi_{C_{vk}}(0^d) \leq \varepsilon(d).$$

Since the rows of D' are idenitcal, $\varphi_j(D') \in \{0, 1\}$ for every $j \in [m]$. So for at least a $1 - \varepsilon(d)$ fraction of $j \in [m]$, we have $\varphi_j(D') = 0$.

Let $\widehat{D}' = \mathcal{M}(D')$. By repeating the signature-forging argument, we see that with probability $1 - \mathsf{negl}(d)$

$$\frac{1}{m}\sum_{j=1}^{m} \varphi_j(\widehat{D}') \leq \varepsilon(d)$$

and thus there must exist a subset $J' \subseteq [m]$ of size $|J'| \geq 2m/3$ such that $j \in J' \implies \varphi_j(\widehat{D}') \leq 3\varepsilon(d) \leq \mathsf{negl}(d)$. So $\varphi_j(\widehat{D}') = 0$ for every $j \in J'$ as well. There must also exist a set $J'' \subseteq J'$ of size $|J''| = (2/3 - \varepsilon(d))m$, such that for every $j \in J''$, $\varphi_j(D') = \varphi_j(\widehat{D}') = 0$. So if $\widehat{D}'$ is $1/3$-accurate for $\mathcal{Q}$ as relaxed synthetic data it must be that $f(0^{\widehat{n}}, \varphi_j) \leq 1/3$ for every $j \in J''$.

By our choice of J and J'' there must exist $j \in J \cap J''$ such that:

1. $f(0^{\widehat{n}}, \varphi_j) \geq 2/3$, and
2. $f(0^{\widehat{n}}, \varphi_j) \leq 1/3$,

which is a contradiction.

□

4.4.4 Positive Results for Relaxed Synthetic Data

In this section we present an efficient, accurate sanitizer for small (e.g. polynomial in d) families of parity queries that outputs symmetric relaxed synthetic data and show that this sanitizer also yields accurate answers for any family of constant-arity predicates when evaluated as a relaxed synthetic data. Our result for parities shows that relaxed synthetic data (and even symmetric relaxed synthetic data) allows for more efficient sanitization than standard synthetic data, since Theorem 4.16 rules out an accurate, efficient sanitizer that produces a standard synthetic database, even for the class of 3-literal parity predicates. Our result for parities also shows that our hardness result for symmetric relaxed synthetic data (Theorem 4.26) is tight with respect to the required hardness of approximation, since the class of 3-literal parity predicates is $(1/2-\varepsilon)$-hard-to-approximate [47]

A function $f : \{0,1\}^d \to \{0,1\}$ is a *k-junta* if it depends on at most k variables. Let $\mathcal{J}_{d,k}$ be the set of all k-juntas on d variables.

Theorem 4.28. *There exists an ϵ-differentially private sanitizer that runs in time $\mathrm{poly}(n, d)$ and produces relaxed synthetic data and is (α, β)-accurate for $\mathcal{J}_{d,k}$ when*

$$n \geq \frac{C\binom{d}{\leq k} \log\left(\binom{d}{\leq k}/\beta\right)}{\alpha\epsilon}$$

for a sufficiently large constant C, where $\binom{d}{\leq k} = \sum_{i=0}^{k} \binom{d}{i}$.

The privacy, accuracy, and efficiency guarantees of our theorem can be achieved without relaxed synthetic data simply by releasing a vector of noisy answers to the queries [30]. Our sanitizer will, in fact, begin with this vector of noisy answers and construct relaxed synthetic data from those answers. Our technique is similar to that of Barak et. al. [8], which begins with a vector of noisy answers to *parity queries* (defined in Section 4.4.4) and constructs a (standard) synthetic database that gives answers to each query that are close to the initial noisy answers. They construct their synthetic database by solving a linear program over 2^d variables that correspond to the frequency of each possible row $x \in \{0,1\}^d$, and thus their sanitizer runs in time exponential in d. Our sanitizer also starts with a vector of noisy answers to parity queries and *efficiently* constructs symmetric relaxed synthetic data that gives answers to each query that are close to the initial noisy answers after applying a *fixed linear scaling*. We then show that the database our sanitizer constructs is also accurate for the family of k-juntas using an affine shift that depends on the junta.

Efficient Sanitizer for Parities

To prove Theorem 4.28, we start with a sanitizer for *parity predicates*.

Definition 4.29 (Parity Predicate)**.** A function $\chi : \{0,1\}^d \to \{-1,1\}$ is a *parity predicate*[10] if there exists a vector $s \in \{0,1\}^d$ s.t.

$$\chi(x) = \chi_s(x) = (-1)^{\langle x,s \rangle}.$$

We will use $wt(s) = \sum_{i=1}^{d} s_i$ to denote the number of non-zero entries in s.

[10] In the preliminaries we define a predicate to be a $\{0,1\}$-valued function but our definition naturally generalizes to $\{-1,1\}$-valued functions. For $q : \{0,1\}^d \to \{-1,1\}$ and database $D = (x^{(1)}, \ldots, x^{(n)}) \in (\{0,1\}^d)^n$, we define $q(D) = \frac{1}{n}\sum_{i=1}^{n} q(x^{(i)})$

Theorem 4.30. *Let $\mathcal{P}$ be a family of parity predicates on d variables such that $\chi_{0^d} \notin \mathcal{P}$. There exists an ϵ-differentially private sanitizer $\mathcal{M}(D, \mathcal{P})$ that runs in time $\mathrm{poly}(n, d)$ and produces symmetric relaxed synthetic data that is (α, β)-accurate for $\mathcal{P}$ when*

$$n \geq \frac{2|\mathcal{P}| \log\left(2|\mathcal{P}|/\beta\right)}{\alpha\epsilon}.$$

The analysis of our sanitizer will make use of the following standard fact about parity predicates

Fact 4.31. *Two parity predicates $\chi_s, \chi_t : \{0,1\}^d \to \{-1,1\}$ are either identical or orthogonal. Specifically, for $s \neq t$, $s \neq 0^d$ and $b \in \{-1,1\}$,*

$$\mathop{\mathbb{E}}_{x \leftarrow_R \{0,1\}^d}[\chi_s(x) | \chi_t(x) = b] = \mathop{\mathbb{E}}_{x \leftarrow_R \{0,1\}^d}[\chi_s(x)] = 0.$$

Our sanitizer will start with noisy answers to the predicate queries $\chi_s(D)$. Each noisy answer will be the true answer perturbed with noise from a *Laplace distribution*. The Laplace distribution $Lap(\sigma)$ is a continuous distribution on $\mathbb{R}$ with probability density function $p_\sigma(y) \propto \exp(-|y|/\sigma)$. The following theorem of Dwork, et. al. [30] shows that these queries are differentially private for an appropriate choice of σ.

Theorem 4.32 ([30])**.** *Let $(q_1, q_2, \ldots, q_k)$ be a set of predicates and let $\sigma = k/n\varepsilon$ and let $D \in (\{0,1\}^d)^n$ be a database. Then the mechanism $\mathcal{M}(D) = (q_1(D) + Z_1, q_2(D) + Z_2, \ldots, q_k(D) + Z_k)$, where $(Z_1, \ldots, Z_k)$ are independent samples from $Lap(\sigma)$ is ϵ-differentially private.*

In order to argue about the accuracy of our mechanism we need to know how much error is introduced by noise from the Laplace distribution. The following fact gives a bound on the tail of a Laplace random variable.

Fact 4.33. *The tail of the Laplace distribution decays exponentially. Specifically,*

$$\Pr[|Lap(\sigma)| \geq t] = \exp(-t/\sigma).$$

Now we present our sanitizer for queries that are parity functions. We will not consider the query χ_{0^d} as $\chi_{0^d}(x) = 1$ for every $x \in \{0,1\}^d$. Let $\mathcal{P}$ be a set of parity functions that does not contain χ_{0^d}. We now present a $\mathrm{poly}(n, d, |\mathcal{P}|)$-time sanitizer for $\mathcal{P}$.

Our sanitizer starts by getting noisy estimates of the quantities $\chi(D)$ for each predicate $\chi \in \mathcal{P}$ by adding Laplace noise. Then it builds the relaxed synthetic data $\widehat{D}$ in blocks of rows. Each block

of rows is "assigned" to contain an answer to a query χ. In that block we randomly choose rows such that the expected value of χ on each row equals the noisy estimate of $\chi(D)$. By Fact 4.31, the expected value of every other predicate χ' is 0 for rows in this block. The sanitizer is accurate so long as the total number of rows is sufficient for the value of $\chi(\widehat{D})$ to be concentrated around its expectation.

Sanitizer $\mathcal{M}(D, \mathcal{P})$, where $P = \{\chi^{(1)}, \ldots, \chi^{(t)}\}$:

Let $\sigma := |\mathcal{P}|/n\epsilon \quad T := (2|\mathcal{P}|/\alpha^2)\log(4|\mathcal{P}|/\beta)$

For: $j = 1, \ldots, t$

Let $a_j := \chi^{(j)}(D) + \mathrm{Lap}(\sigma)$

For: $i = jT + 1$ to $(j+1)T$

With probability $(a_j + 1)/2$: Let $\widehat{x}^{(i)} \leftarrow_{\mathrm{R}} \left\{x \in \{0,1\}^d \mid \chi^{(j)}(x) = 1\right\}$

Otherwise: Let $\widehat{x}^{(i)} \leftarrow_{\mathrm{R}} \left\{x \in \{0,1\}^d \mid \chi^{(j)}(x) = -1\right\}$

Return: $\widehat{D} = (\widehat{x}^{(1)}, \ldots, \widehat{x}^{(tT)})$

Evaluator $\mathcal{E}_{\mathcal{P}}(\widehat{D}, \chi)$:

Return: $|\mathcal{P}| \cdot \chi(\widehat{D})$

Figure 11: An efficient one-shot sanitizer for juntas generating relaxed synthetic data.

The following claims will suffice to establish Theorem 4.30

Claim 4.34. *$\mathcal{M}$ is ϵ-differentially private.*

Proof. The output of $\mathcal{M}$ only depends on the answer to $|\mathcal{P}|$ predicate queries. By Theorem 4.32 the answers to $|\mathcal{P}|$ predicate queries perturbed by independent samples from $Lap(|\mathcal{P}|/n\epsilon)$ is ϵ-differentially private. □

Claim 4.35. *$\mathcal{M}$ is (α, β)-accurate for $\mathcal{P}$ when*

$$n \geq \frac{2|\mathcal{P}|\log(2|\mathcal{P}|/\beta)}{\alpha\epsilon}.$$

Proof. We want to show that for every $\chi^{(j)} \in \mathcal{P}$

$$\left||\mathcal{P}| \cdot \chi^{(j)}(\widehat{D}) - \chi^{(j)}(D)\right| \leq \alpha$$

except with probability β. To do so we consider separately the error introduced in going from $\chi^{(j)}(D)$ to a_j using Laplacian noise and the error introduced in going from noisy answers a_j to $\chi^{(j)}(\widehat{D})$ by sampling rows at random.

First we bound the error introduced by the noisy queries to D. Specifically, we want to show that for every $\chi^{(j)} \in \mathcal{P}$

$$\left|\chi^{(j)}(D) - a_j\right| \leq \alpha/2$$

except with probability $\beta/2$. For each $\chi^{(j)}$ we have

$$\Pr[|\chi^{(j)}(D) - a_j| \geq \alpha/2] \leq \exp(-n\epsilon\alpha/2|\mathcal{P}|)$$

by Fact 4.33. So by a union bound we have

$$\Pr[\exists\chi^{(j)}\ |\chi^{(j)}(D) - a_j| \geq \alpha/2] \leq |\mathcal{P}|\exp(-\alpha/2\sigma) \leq |\mathcal{P}|\exp(-n\epsilon\alpha/2|\mathcal{P}|) < \beta/2,$$

so long as

$$n \geq \frac{2|\mathcal{P}|\log(2|\mathcal{P}|/\beta)}{\alpha\epsilon}.$$

We also want to show that for every $\chi^{(j)} \in \mathcal{P}$

$$\left||\mathcal{P}| \cdot \chi^{(j)}(\widehat{D}) - a_j\right| \leq \alpha/2$$

except with probability $\beta/2$, where a_j is the noisy answer for $\chi^{(j)}(D)$ computed in $\mathcal{M}(D)$. To do so, we will show that the expectation of $|\mathcal{P}|\chi^{(j)}(\widehat{D})$ is indeed a_j, then we will use a Chernoff-Hoeffding bound to show that the random rows generated by $\mathcal{M}(D)$ are close to their expectation. Finally we take a union bound over all $\chi \in \mathcal{P}$.

Fix $\chi^{(j)} \in \mathcal{P}$ and consider $\chi^{(j)}(\widehat{D})$. $\chi^{(j)}(\widehat{D})$ is the sum of T independent biased coin flips. In rows $jT+1, jT+2, \ldots, (j+1)T$ (the rows where we focus on $\chi^{(j)}$) the expectation of each coin flip is a_j, and in all other rows the expectation of each coin flip is 0 by Fact 4.31. Thus

$$\mathbb{E}\left[\chi^{(j)}(\widehat{D})\right] = \mathbb{E}\left[\frac{1}{\widehat{n}}\sum_{i=1}^{\widehat{n}}\chi^{(j)}(\widehat{x}^{(i)})\right] = a_j/|\mathcal{P}|$$

for every $\chi^{(j)} \in \mathcal{P}$.

By a Chernoff-Hoeffding Bound[11] we conclude

$$\Pr\left[\left||\mathcal{P}| \cdot \chi^{(j)}(\widehat{D}) - a_j\right| \geq \alpha/2\right] < 2\exp(-T\alpha^2/2|\mathcal{P}|).$$

[11]One form of the Chernoff-Hoeffding Bound states if $X_1, \ldots, X_n$ are independent random variables over $[0,1]$ and $X = (1/n)\sum_{i=1}^{n} X_i$ then $\Pr[|X - \mathbb{E}[X]| \geq t] < 2\exp(-nt^2/2)$

By taking a union bound over $\mathcal{P}$ we conclude

$$\Pr\left[\exists \chi^{(j)} \ \left||\mathcal{P}\cdot|\chi^{(j)}(\widehat{D}) - a_j\right| \geq \alpha/2\right] < 2|\mathcal{P}|\exp(-T\alpha^2/2|\mathcal{P}|) \leq \beta/2.$$

Combining the two bounds suffices to prove the claim. □

Efficient Sanitizer for k-Juntas

We now show that our sanitizer for parity queries can also be used to give accurate answers for any family of k-juntas, for constant k. We start with the observation that k-juntas only have Fourier mass on coefficients of weight at most k. Alternatively, this says that any k-junta can be written as a linear combination of parity functions on at most k variables. (In the language of our previous construction, χ_s such that $wt(s) \leq k$.) Thus we start by running our sanitizer for parity predicates on the set $\mathcal{P}_k$ containing all parity predicates on at most k variables. We have to modify the evaluator function to take into account that not every k-junta predicate has the same bias. Indeed, we cannot control $\chi_{0^d}(\widehat{D})$ in our output, as $\chi_{0^d}(D) = 1$ for any database. Thus our evaluator will apply an affine shift to each result that depends on the junta. Because the evaluator depends on the predicate, the resulting sanitizer no longer outputs symmetric relaxed synthetic data.

The use of a sanitizer for parity queries as a building block to construct a sanitizer for arbitrary k-juntas is inspired by [8], which uses a noisy vector of answers to parity queries as a building block to construct synthetic data for a particular class of k-juntas (conjunctions on k-literals). However, while their sanitizer constructs a standard synthetic database and is inefficient, our construction of symmetric relaxed synthetic data for parity predicates is efficient, and thus our eventual sanitizer for k-juntas will also be efficient.

Consider a predicate $c : \{0,1\}^d \to \{0,1\}$. Then we can take the Fourier expansion

$$c(x) = \sum_{s\in\{0,1\}^d} \widehat{c}(s)\chi_s(x)$$

where

$$\widehat{c}(s) = \mathop{\mathbb{E}}_{x\leftarrow_R\{0,1\}^d}[c(x)\chi_s(x)].$$

The accuracy of our sanitizer relies on the following fact about the Fourier coefficients of k-juntas

Fact 4.36. *If $c : \{0,1\}^d \to \{0,1\}$ is a k-junta, then $wt(s) > k \implies \widehat{c}(s) = 0$*

Let $\mathcal{P}_k = \left\{\chi_s \mid s \in \{0,1\}^d, 1 \leq wt(s) \leq k\right\}$. Our sanitizer for k-juntas is just the sanitizer for parities applied to the set $\mathcal{P}_k$. We now define the evaluator that computes the answer to conjunction queries from the output of $\mathcal{M}(D, \mathcal{P}_k)$.

Evaluator $\mathcal{E}(\widehat{D}, q)$ for a k-junta q:

return $|\mathcal{P}_k| q(\widehat{D}) - (|\mathcal{P}_k| - 1)\, \widehat{q}(\emptyset)$

Efficiency and privacy follow from the analysis of $\mathcal{M}$. Let $\mathcal{J}_{d,k}$ be a family of all k-juntas on d variables.

Theorem 4.37. *$\mathcal{M}(D, \mathcal{P}_k)$ is (α, β)-accurate for $\mathcal{J}_{d,k}$ using $\mathcal{E}$ when*

$$n \geq \frac{2|\mathcal{P}_k| \log(2|\mathcal{P}_k|/\beta)}{\alpha\epsilon}.$$

Proof. Let $c \in \mathcal{J}_{d,k}$ be a fixed predicate. Assume that $\widehat{D}$ is α-accurate for $\mathcal{P}_k$ using $\mathcal{E}_{\mathcal{P}}$. This event occurs with probability at least $1 - \beta$ by Theorem 4.30 and our assumption on n. We now analyze the quantity $c(\widehat{D})$.

$$\begin{aligned}
q(\widehat{D}) &= \frac{1}{\widehat{n}} \sum_{i=1}^{\widehat{n}} q(\widehat{x}^{(i)}) \\
&= \frac{1}{\widehat{n}} \sum_{i=1}^{\widehat{n}} \sum_{s \in \{0,1\}^d} \widehat{q}(s) \chi_s(\widehat{x}^{(i)}) \\
&= \sum_{s \in \{0,1\}^d : |s| \leq k} \widehat{q}(s) \chi_s(\widehat{D}) && (4.3) \\
&= \widehat{q}(\emptyset) + \sum_{s \in \{0,1\}^d : 1 \leq |s| \leq k} \widehat{q}(s) \chi_s(\widehat{D}) && (4.4) \\
&\leq \widehat{q}(\emptyset) + \sum_{s \in \{0,1\}^d : 1 \leq |s| \leq k} \widehat{q}(s) \left(\frac{\chi_s(D) + \alpha}{|\mathcal{P}_k|} \right) && (4.5) \\
&\leq \frac{1}{|\mathcal{P}_k|} \left((|\mathcal{P}_k| - 1)\, \widehat{q}(\emptyset) + \sum_{s \in \{0,1\}^d : |s| \leq k} (\widehat{q}(s) \chi_s(D) + \alpha) \right) \\
&= \frac{1}{|\mathcal{P}_k|} \left((|\mathcal{P}_k - 1)\, \widehat{q}(\emptyset) + q(D) \right) + \alpha
\end{aligned}$$

where step 4.3 uses Fact 4.36, step 4.4 uses the fact that $\chi_{0^d}(x) = 1$ everywhere, and step 4.5 uses

the fact that $\widehat{D}$ is α-accurate for $\mathcal{P}_k$ when evaluated by $\mathcal{E}_{\mathcal{P}}$. A similar argument shows that

$$q(\widehat{D}) \geq \frac{1}{|\mathcal{P}_k|}\left(\left(|\mathcal{P}_k| - 1\right)\widehat{q}(\emptyset) + q(D)\right) - \alpha$$

Thus $\widehat{D} = \mathcal{M}(D)$ is α-accurate for $\mathcal{J}_{d,k}$ using $\mathcal{E}$ with probability at least $1 - \beta$.

□

Chapter 5

Faster Algorithms for Privately Releasing Marginal Queries

In this chapter and the next, we continue our study of marginal queries, one of the most important classes of statistics on a dataset. In Chapter 4 we discussed some of the prior work on differentially private release of marginal queries. However, we will review that background so that the next two chapters will be reasonably self-contained.

Recall that a marginal query is specified by a set $S \subseteq [d]$ and a pattern $t \in \{0,1\}^{|S|}$. The query asks, "What fraction of the individual records in D has each of the attributes $j \in S$ set to t_j?" Designing *efficient* one-shot sanitizers for marginals has been identified as a major open problem in differential privacy (cf. [8]). That is, we would like an efficient algorithm to create a differentially private summary of the database that enables analysts to answer each of the 3^d marginal queries up to some small additive error, say $\pm.01$. A natural subclass of marginals are *k-way marginals*, the subset of marginals specified by sets $S \subseteq [d]$ such that $|S| \leq k$.

As we've discussed in Chapter 2, if we perturb the answers to every marginal query with independent noise, then we can release the answers to every k-way marginal query with non-trivial accuracy as long as $|D| \gtrsim d^{\Theta(k)}$. However, it may not be practical to collect enough data to ensure that this condition holds. Even if we do have a sufficiently large database, it would require time $d^{\Theta(k)}$ just to compute the answers to every k-way marginal query, which may be prohibitive. Fortunately, algorithms such as private multiplicative weights show that it is possible to privately answer all k-way marginals as long as $|D| \geq \widetilde{\Theta}(k\sqrt{d})$. Unfortunately, all of these algorithms

have running time at least 2^d, even when $|\mathcal{Q}|$ is the set of 2-way marginals. As we have shown in Chapter 4, this running time is inherent for algorithms such as these that generate private synthetic data. In fact, under a strong hardness assumption, even releasing a synthetic database that is accurate for just the set of all 2-way marginals requires time nearly 2^d.

Given this state of affairs, it is natural to seek efficient one-shot sanitizers that do not generate synthetic data capable of privately releasing approximate answers to marginal queries even when $|D| \ll d^k$. A series of works [41, 22, 45] have shown how to construct such one-shot sanitizers for k-way marginal queries with small *average error* (over various distributions on the queries) with both running time and minimum database size much smaller than d^k (e.g. $d^{O(1)}$ for product distributions [41, 22] and $\min\{d^{O(\sqrt{k})}, d^{O(d^{1/3})}\}$ for arbitrary distributions [45]). Hardt et al. [45] also gave an algorithm for privately releasing k-way marginal queries with small *worst-case error* and minimum database size much smaller than d^k. However the running time of their algorithm is still $d^{\Theta(k)}$, which is polynomial in the number of queries.

In this chapter, we give the first algorithms capable of releasing k-way marginals up to small worst-case error with both running time and minimum database size substantially smaller than d^k. Specifically, we show how to create a private summary in time $d^{O(\sqrt{k})}$ that gives approximate answers to all k-way marginals as long as $|D|$ is at least $d^{O(\sqrt{k})}$. When $k = d$, our algorithm runs in time $2^{\widetilde{O}(\sqrt{d})}$, and is the first algorithm for releasing *all* marginals in time $2^{o(d)}$ (and thus the first to overcome the "synthetic data barrier."

5.1 Our Results and Techniques

In this chapter, we present faster algorithms for releasing marginals and other classes of counting queries.

Theorem 5.1 (Releasing Marginals)**.** *There exists a constant C such that for every $k, d, n \in \mathbb{N}$ with $k \leq d$, every $\alpha \in (0, 1]$, and every $\varepsilon > 0$, there is an ε-differentially private one-shot sanitizer that, on input a database $D \in (\{0,1\}^d)^n$, runs in time $|D| \cdot d^{C\sqrt{k}\log(1/\alpha)}$ and releases a summary that enables computing each of the k-way marginal queries on D up to an additive error of at most α, provided that $|D| \geq d^{C\sqrt{k}\log(1/\alpha)}/\varepsilon$.*

For notational convenience, we focus on *monotone k-way disjunction queries*. However, our results extend straightforwardly to general non-monotone k-way disjunction queries (see Sec-

tion 5.4.1), which are equivalent to k-way marginals. A monotone k-way disjunction is specified by a set $S \subseteq [d]$ of size k and asks what fraction of records in D have at least one of the attributes in S set to 1.

Our algorithm is inspired by a series of works reducing the problem of private query release to various problems in learning theory [41, 22, 45, 38]. One ingredient in this line of work is a shift in perspective introduced by Gupta, Hardt, Roth, and Ullman [41]. Instead of viewing disjunction queries as a set of functions on the database, they view the database as a function $f_D \colon \{0,1\}^d \to [0,1]$, in which each vector $s \in \{0,1\}^d$ is interpreted as the indicator vector of a set $S \subseteq [d]$, and $f_D(s)$ equals the evaluation of the disjunction specified by S on the database D. They use the structure of the functions f_D to privately learn an approximation g_D that has small *average error* over any product distribution on disjunctions.[12]

Cheraghchi, Klivans, Kothari, and Lee [22] observed that the functions f_D can be approximated by a low-degree polynomial with small average error over the uniform distribution on disjunctions. They then use a private learning algorithm for low-degree polynomials to release an approximation to f_D; and thereby obtain an improved dependence on the accuracy parameter, as compared to [41].

Hardt, Rothblum, and Servedio [45] observe that f_D is itself an average of disjunctions (each row of D specifies a disjunction of bits in the indicator vector $s \in \{0,1\}^d$ of the query), and thus develop private learning algorithms for threshold of sums of disjunctions. These learning algorithms are also based on low-degree approximations of sums of disjunctions.

They show how to use their private learning algorithms to obtain a sanitizer with small average error over *arbitrary distributions* with running time and minimum database size $d^{O(\sqrt{k})}$. They then are able to apply the private boosting technique of Dwork, Rothblum, and Vadhan [35] to obtain worst-case accuracy guarantees. Unfortunately, the boosting step incurs a blowup of d^k in the running time.

We improve the above results by showing how to *directly* compute (a noisy version of) a polynomial p_D that is privacy-preserving and still approximates f_D on *all* k-way disjunctions, as long as $|D|$ is sufficiently large. Specifically, the running time and the database size requirement of our algorithm are both polynomial in the number of monomials in p_D, which is $d^{O(\sqrt{k})}$. By "directly", we mean that we compute p_D from the database D itself and perturb its coefficients, rather

[12] In their learning algorithm, privacy is defined with respect to the rows of the database D that defines f_D, not with respect to the examples given to the learning algorithm (unlike earlier works on "private learning" [51]).

than using a learning algorithm. Our construction of the polynomial p_D uses the same low-degree approximations exploited by Hardt et al. in the development of their private learning algorithms.

In summary, the main difference between prior work and ours is that prior work used learning algorithms that have restricted access to the database, and released the hypothesis output by the learning algorithm. In contrast, we do not make use of any learning algorithms, and give our release algorithm direct access to the database. by doing so, we enable our algorithm to achieve a worst-case error guarantee while maintaining a minimal database size and running time much smaller than the size of the query set. Our algorithm is also substantially simpler than that of Hardt et al.

Our approach extends beyond approximations by polynomials. More generally, suppose there is a *feature space* of functions $\mathcal{S}$ such that for every database D, f_D can be approximated by a linear combination of functions in $\mathcal{S}$ with bounded coefficients. Then there is an algorithm that releases an approximation to f_D with running time and minimum database size polynomial in $|\mathcal{S}|$. In the case where we approximate f_D by a polynomial of degree t, the feature space would consist of all monomials of total degree at most t.

We also consider other families of counting queries. We define the class of *r-of-k queries*. Like a monotone k-way disjunction, an r-of-k query is defined by a set $S \subseteq [d]$ such that $|S| \leq k$. The query asks what fraction of the rows of D have at least r of the attributes in S set to 1. For $r = 1$, these queries are exactly monotone k-way disjunctions, and r-of-k queries are a strict generalization.

Theorem 5.2 (Releasing r-of-k Queries)**.** *For every $r, k, d, n \in \mathbb{N}$ with $r \leq k \leq d$, every $\alpha \in (0, 1]$, and every $\varepsilon > 0$ there is an ε-differentially private one-shot sanitizer that, on input a database $D \in (\{0,1\}^d)^n$, runs in time $|D| \cdot d^{\widetilde{O}\left(\sqrt{rk \log(1/\alpha)}\right)}$ and releases a summary that enables computing each of the r-of-k queries on D up to an additive error of at most α, provided that $|D| \geq d^{\widetilde{O}\left(\sqrt{rk \log(1/\alpha)}\right)}/\varepsilon$.*

Since monotone k-way disjunctions are just r-of-k queries where $r = 1$, thus Theorem 5.2 implies a release algorithm for disjunctions with quadratically better dependence on $\log(1/\alpha)$, at the cost of slightly worse dependence on k (implicit in the switch from $O(\cdot)$ to $\widetilde{O}(\cdot)$).

Finally, we give a sanitizer for privately releasing databases in which the *rows* of the database are interpreted as decision lists, and the *queries* are inputs to the decision lists. That is, instead of each record in D being a string of d attributes, each record is an element of the set $\mathrm{DL}_{k,m}$,

which consists of all length-k decision lists over m input variables. (See Section 5.4.3 for a precise definition.) A query is specified by a string $y \in \{0,1\}^d$ and asks "What fraction of database participants would make a certain decision based on the input y?"

As an example application, consider a database that allows high school students to express their preferences for colleges in the form of a decision list. For example, a student may say, "If the school is ranked in the top ten nationwide, I am willing to apply to it. Otherwise, if the school is rural, I am unwilling to apply. Otherwise, if the school has a good basketball team then I am willing to apply to it." And so on. Each student is allowed to use up to k attributes out of a set of m binary attributes. Our sanitizer allows any college (represented by its m binary attributes) to determine the fraction of students willing to apply.

Theorem 5.3 (Releasing Decision Lists)**.** *For any $k, m \in \mathbb{N}$ s.t. $k \leq m$, any $\alpha \in (0,1]$, and any $\varepsilon > 1/n$, there is an ε-differentially private one-shot sanitizer with running time $m^{\widetilde{O}\left(\sqrt{k}\log(1/\alpha)\right)}$ that, on input a database $D \in (\mathrm{DL}_{k,m})^n$, releases a summary that enables computing any length-k decision list query up to an additive error of at most α on every query, provided that $|D| \geq m^{\widetilde{O}\left(\sqrt{k}\log(1/\alpha)\right)}/\varepsilon$.*

For comparison, we note that all the results on releasing k-way disjunctions (including ours) also apply to a dual setting where the database *records* specify a k-way disjunction over m bits and the *queries* are m-bit strings (in this setting m plays the role of d). Theorem 5.3 generalizes this dual version of Theorem 5.1, as length-k decision lists are a strict generalization of k-way disjunctions.

We prove the latter two results (Theorems 5.2 and 5.3) using the same approach outlined for marginals (Theorem 5.1), but with different low-degree polynomial approximations appropriate for the different types of queries.

See Table 1 for a summary of the prior results on differentially private release of k-way marginal queries. In that table, the database size column indicates the minimum database size required to release answers to k-way marginals up to an additive error of α. For clarity, Table 1 ignores the dependence on the privacy parameters and the failure probability of the algorithms.

Table 1: Summary of results on differentially private release of k-way marginals.

Paper	Running Time	Database Size	Error Type	Synth. Data
[26, 34, 15, 30]	$d^{O(k)}$	$O(d^{k/2}/\alpha)$	Worst case	N
[8]	$2^{O(d)}$	$O(d^{k/2}/\alpha)$	Worst case	Y
[16, 32, 35, 43]	$2^{O(d)}$	$\widetilde{O}(k\sqrt{d}/\alpha^2)$	Worst case	Y
[41]	$d^{\widetilde{O}(1/\alpha^2)}$	$d^{\widetilde{O}(1/\alpha^2)}$	Product Dists.	N
[38]	$\widetilde{O}(d^2/\alpha^{10})$	$\widetilde{O}(d/\alpha^6)$	Uniform Dist.	N
[38]	$d \cdot 2^{\log^2(1/\alpha)}$	$d^2 \cdot 2^{\log^2(1/\alpha)}$	Uniform Dist.	Y
[45]	$d^{O(d^{1/3}\log(1/\alpha))}$	$d^{O(d^{1/3}\log(1/\alpha))}$	Any Dist.	N
[45]	$d^{O(k)}$	$d^{O(d^{1/3}\log(1/\alpha))}$	Worst case	N
[45]	$d^{O(\sqrt{k}\log(1/\alpha))}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Any Dist.	N
[45]	$d^{O(k)}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Worst case	N
This chapter	$d^{O(\sqrt{k}\log(1/\alpha))}$	$d^{O(\sqrt{k}\log(1/\alpha))}$	Worst case	N

5.2 Preliminaries

5.2.1 The Vector-Valued Laplace Mechanism

In Chapter 2 we described the Laplace mechanism as a technique for answering real-valued queries. For the results of this chapter it will be more convenient to apply the Laplace mechanism to a *vector-valued* query. In cases where the query is vector-valued, we can apply the Laplace mechanism to each coordinate individually. Suppose $f\colon (\{0,1\}^d)^n \to \mathbb{R}^m$ has low *(global)* L_∞*-sensitivity*, which we define to be

$$\Delta_f = \max_{D \sim D'} \|f(D) - f(D')\|_\infty.$$

Let $\mathrm{Lap}^m(\sigma)$ denote the m-variate probability distribution that arises when each coordinate is chosen independently according to $\mathrm{Lap}(\sigma)$. We have the following lemma, which is a corollary of Lemma 2.9.

Lemma 5.4. *Let $f : (\{0,1\}^d)^n \to \mathbb{R}^m$ be a query with L_∞-sensitivity Δ_f and let $\mathcal{M}_{\mathrm{Lap}}(D, f) = f(D) + Z = a$, where $Z \leftarrow_R \mathrm{Lap}^m(\sigma)$ for a parameter $\sigma > 0$. Then the following both hold:*

1. *For any $\varepsilon > 0$, if Z is chosen according to $\mathrm{Lap}^m(\Delta_f m/\varepsilon)$, then $\mathcal{M}_{\mathrm{Lap}}$ is $(\varepsilon, 0)$-differentially*

private. Furthermore, for any $\beta > 0$, with probability at least $1 - \beta$,

$$\|a - f(D)\|_1 \leq \frac{\Delta_f m^2 \log(m/\beta)}{\varepsilon}.$$

2. *For any $\varepsilon, \delta > 0$, if Z is chosen according to $\mathrm{Lap}^m(\Delta_f \sqrt{8m \log(1/\delta)}/\varepsilon)$, then $\mathcal{M}_{\mathrm{Lap}}$ is (ε, δ)-differentially private. Furthermore, for any $\beta > 0$, with probability at least $1 - \beta$,*

$$\|a - f(D)\|_1 \leq \frac{\Delta_f m^{3/2} \sqrt{8 \log(1/\delta)} \log(m/\beta)}{\varepsilon}.$$

Recall from Chapter 2 that we are assuming that noise from the Laplace distribution is replaced with noise from a similar discrete distribution, and that a sample from this distribution can be drawn in time $\mathrm{poly}(\sigma)$. Thus the running time of the Laplace mechanism as used in Lemma 5.4 is $\mathrm{poly}(\sigma) = \mathrm{poly}(\Delta_f, m, 1/\varepsilon, \log(1/\delta), \log(1/\beta))$.

5.2.2 Query Function Families

We take the approach of Gupta et al. [41] and think of the database D as specifying a function f_D mapping queries q to their answers $q(D)$, which we call the *$\mathcal{Q}$-representation of D*. We now describe this transformation more formally:

Definition 5.5 ($\mathcal{Q}$-Function Family)**.** Let $\mathcal{Q} = \{q_y\}_{y \in Y_{\mathcal{Q}} \subseteq \{0,1\}^m}$ be a set of counting queries on a data universe $\mathcal{X}$, where each query is indexed by an m-bit string. We define the *index set of $\mathcal{Q}$* to be the set $Y_{\mathcal{Q}} = \{y \in \{0,1\}^m \mid q_y \in \mathcal{Q}\}$.

We define the *$\mathcal{Q}$-function family* $\mathcal{F}_{\mathcal{Q}} = \{f_x : \{0,1\}^m \to \{0,1\}\}_{x \in \mathcal{X}}$ as follows: For every possible database row $x \in \mathcal{X}$, the function $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ is defined as $f_{\mathcal{Q},x}(y) = q_y(x)$. Given a database $D \in \mathcal{X}^n$ we define the function $f_{\mathcal{Q},D} : \{0,1\}^m \to [0,1]$ where $f_{\mathcal{Q},D}(q) = \frac{1}{n} \sum_{i=1}^n f_{\mathcal{Q},x^{(i)}}(q)$. When $\mathcal{Q}$ is clear from context we will drop the subscript $\mathcal{Q}$ and simply write f_x, f_D, and $\mathcal{F}$.

For some intuition about this transformation, when the queries are monotone k-way disjunctions on a database $D \in (\{0,1\}^d)^n$, the queries are defined by sets $S \subseteq [d]$, $|S| \leq k$. In this case each query can be represented by the d-bit indicator vector of the set S, with at most k non-zero entries. Thus we can take $m = d$ and $Y_{\mathcal{Q}} = \left\{y \in \{0,1\}^d \mid \sum_{j=1}^d y_j \leq k\right\}$.

5.2.3 Low-Dimensional Linear Approximations

Let $\mathcal{S} = \{s\colon \{0,1\}^m \to [0,1]\}$ be a *feature space* of efficiently computable (time $\mathrm{poly}(m)$) m-variate functions. For a vector $\vec{c} = (c_s)_{s\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$, let $g_{\vec{c}}$ be the function

$$g_{\vec{c}}(y) = \sum_{s\in\mathcal{S}} c_s \cdot s(y).$$

Abusing notation, we will often associate the function $g_{\vec{c}}$ with its coefficient vector $\vec{c}$ and use $\vec{c}(y)$ in place of $g_{\vec{c}}(y)$. Notice that computing $\vec{c}(y)$ is equivalent to computing $\vec{c}\cdot\vec{y}$, where $\vec{y} \in [0,1]^{|\mathcal{S}|}$ is the vector $(y_s)_{s\in\mathcal{S}}$ formed by taking $\vec{y}_s = s(y)$ for every $s \in \mathcal{S}$.

In many cases, the function $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ can be approximated well on all the indices in $Y_{\mathcal{Q}}$ by a linear combination of functions in $\mathcal{S}$, where the linear combination has bounded coefficients. Formally:

Definition 5.6 (Uniform Approximation by a Feature Space)**.** Given a family of m-variate functions $\mathcal{F} = \{f_x\}_{x\in\mathcal{X}}$ and a set $Y \subseteq \{0,1\}^m$, we say that the feature space $\mathcal{S}$ *uniformly γ-approximates $\mathcal{F}$ on Y with (L_∞-)norm T* if for every $x \in \mathcal{X}$, there exists $\vec{c}_x \in [-T,T]^{|\mathcal{S}|}$ such that

$$\forall y \in Y,\ |f_x(y) - \vec{c}_x(y)| \le \gamma.$$

We say that $\mathcal{S}$ *efficiently and uniformly γ-approximates $\mathcal{F}$* if there is an algorithm $\mathcal{S}_{\mathcal{F}}$ that takes $x \in \mathcal{X}$ as input, runs in time $\mathrm{poly}(\log|\mathcal{X}|, |\mathcal{S}|, \log T)$ and outputs a coefficient vector $\vec{c}_x$ such that

$$\forall y \in Y,\ |f_x(y) - \vec{c}_x(y)| \le \gamma.$$

5.3 From Low-Dimensional Approximations to One-Shot Sanitizers

In this section we present a one-shot sanitizer for any family of counting queries $\mathcal{Q}$ such that $\mathcal{F}_{\mathcal{Q}}$ that can be efficiently and uniformly approximated by a feature space $\mathcal{S}$. The algorithm will take an n-row database D and, for each row $x \in D$, constructs a linear combination of functions in $\mathcal{S}$, $\vec{c}_x$ that uniformly approximates the function $f_{\mathcal{Q},x}$ (recall that $f_{\mathcal{Q},x}(q) = q(x)$, for each $q \in \mathcal{Q}$). From these, it constructs a new linear combination $\vec{c}_D = \frac{1}{n}\sum_{x\in D}\vec{c}_x$ that uniformly approximates $f_{\mathcal{Q},D}$.

The final step is to perturb each of the coefficients of $\vec{c}_x$ using noise from a Laplace distribution (Lemma 5.4) and bound the error introduced from the perturbation.

Theorem 5.7 (Releasing Low-Dimensional Approximations)**.** *Let $\mathcal{Q} = \{q_y\}_{y \in Y_\mathcal{Q} \subseteq \{0,1\}^m}$ be a set of counting queries over $\mathcal{X}$, and $\mathcal{F}_\mathcal{Q}$ be the $\mathcal{Q}$ function family (Definition 5.5). Assume that $\mathcal{S}$ efficiently and uniformly γ-approximates $\mathcal{F}_\mathcal{Q}$ on $Y_\mathcal{Q}$ with norm T (Definition 5.6). Then there is a sanitizer $\mathcal{M} \colon \mathcal{X}^n \to \mathbb{R}^{|\mathcal{S}|}$ that*

1. *is ε-differentially private,*
2. *runs in time $\mathrm{poly}(n, \log|\mathcal{X}|, |\mathcal{S}|, \log T, \log(1/\varepsilon))$, and*
3. *is (α, β)-accurate for $\mathcal{Q}$ for*

$$\alpha = \gamma + \frac{4T|\mathcal{S}|^2 \log\left(|\mathcal{S}|/\beta\right)}{\varepsilon n}.$$

Proof. First we construct the sanitizer $\mathcal{M}$. See the relevant codebox below.

Input: A database $D \in \mathcal{X}^n$, an explicit feature space $\mathcal{S}$, and a parameter $\varepsilon > 0$.
For: $i = 1, \ldots, n$
Using efficient approximation of $\mathcal{F}$ by $\mathcal{S}$, compute a coefficient vector $\vec{c}_{x^{(i)}} = \mathcal{S}_\mathcal{F}(x^{(i)})$ that γ-approximates $f_{x^{(i)}}$ on $Y_\mathcal{Q}$.
Let $\vec{c}_D = \frac{1}{n}\sum_{i=1}^n \vec{c}_{x^{(i)}}$, where the sum denotes standard entry-wise vector addition.
Let $\widehat{c}_D = \vec{c}_D + Z$, where Z is drawn from an $|\mathcal{S}|$-dimensional Laplace distribution with parameter $2T/\varepsilon n$ (Section 5.2.1).
Output: $\widehat{c}_D$.

Figure 12: The Sanitizer $\mathcal{M}$

Privacy. We establish that $\mathcal{M}$ is ε-differentially private. This follows from the observation that for any two adjacent $D \sim D'$ that differ only on row i^*,

$$\|\vec{c}_D - \vec{c}_{D'}\|_\infty = \left\|\frac{1}{n}\sum_{i=1}^n \vec{c}_{x^{(i)}} - \frac{1}{n}\sum_{i=1}^n \vec{c}_{x'^{(i)}}\right\|_\infty = \frac{1}{n}\left\|\vec{c}_{x^{(i^*)}} - \vec{c}_{x'^{(i^*)}}\right\|_\infty \leq \frac{2T}{n}.$$

The last inequality is from the fact that for every x, $\vec{c}_x$ is a vector of L_∞ norm at most T. Part 1 of the Theorem now follows directly from the properties of the Laplace Mechanism (Lemma 5.4). Now we construct the evaluator $\mathcal{E}$.

Input: A vector $\widehat{c} \in \mathbb{R}^{|\mathcal{S}|}$ and the description of a query $y \in \{0,1\}^m$.
Output: $\widehat{c}(y)$. Recall that we view $\widehat{c}$ as an m-variate function and $\widehat{c}(y)$ is the evaluation of that function on the point y.

Figure 13: The Evaluator $\mathcal{E}$ for the Sanitizer $\mathcal{M}$.

Efficiency. Next, we show that $\mathcal{M}$ runs in time $\mathrm{poly}(n, \log|\mathcal{X}|, |\mathcal{S}|, \log T, \log(1/\varepsilon))$. Recall that we assumed the polynomial construction algorithm $\mathcal{S}_{\mathcal{F}}$ runs in time $\mathrm{poly}(\log|\mathcal{X}|, |\mathcal{S}|, \log T)$. The algorithm $\mathcal{M}$ needs to run $\mathcal{S}_{\mathcal{F}}$ on each of the n rows, and then it needs to generate a sample from a $|\mathcal{S}|$-dimensional Laplace distribution. This sampling can be done simply by taking $|\mathcal{S}|$ samples from a univariate Laplace distribution with magnitude $\mathrm{poly}(|\mathcal{S}|, T, 1/n, 1/\varepsilon)$. As we have discussed in Section 5.2.1, these samples can be computed in time $\mathrm{poly}(|\mathcal{S}|, T, n, 1/\varepsilon)$. We also establish that $\mathcal{E}$ runs in time $\mathrm{poly}(|\mathcal{S}|, \log T, \log n, \log(1/\varepsilon))$, observe that $\mathcal{E}$ needs to expand the input into an appropriate vector of dimension $|\mathcal{S}|$ and take the inner product with the vector $\widetilde{c}$, whose entries have magnitude $\mathrm{poly}(|\mathcal{S}|, T, 1/n, 1/\varepsilon)$. These observations establish Part 2 of the Theorem.

Accuracy. Finally, we analyze the accuracy of the sanitizer $\mathcal{M}$. First, by the assumption that $\mathcal{S}$ uniformly γ-approximates $\mathcal{F}$ on $Y \subseteq \{0,1\}^m$ with norm T, we have

$$\begin{aligned}\max_{y\in Y} |f_D(y) - \vec{c}_D(y)| &= \max_{y\in Y} \left| \frac{1}{n}\sum_{i=1}^{n} f_{x^{(i)}}(y) - \frac{1}{n}\sum_{i=1}^{n} \vec{c}_{x^{(i)}}(y) \right| \\ &\leq \frac{1}{n}\sum_{i=1}^{n} \max_{y\in Y} |f_{x^{(i)}}(y) - \vec{c}_{x^{(i)}}(y)| \leq \gamma.\end{aligned}$$

Now we want to establish that

$$\Pr\left[\max_{y\in\{0,1\}^m} |\widehat{c}_D(y) - \vec{c}_D(y)| \leq \alpha'\right] \geq 1-\beta$$

for

$$\alpha' = \frac{4T|\mathcal{S}|^2 \log\left(|\mathcal{S}|/\beta\right)}{\varepsilon n},$$

where the probability is taken over the coins of $\mathcal{M}$. Part (3) of the Theorem will then follow by the triangle inequality.

To see that the above statement is true, observe that by the properties of the Laplace mechanism (Lemma 5.4), we have $\Pr\left[\|\widehat{c}_D - \vec{c}_D\|_1 \leq \alpha'\right] \geq 1-\beta$, where the probability is taken over the coins of $\mathcal{M}$. Given that $\|\widehat{c}_D - \vec{c}_D\|_1 \leq \alpha'$, it holds that for every $y \in \{0,1\}^m$,

$$|\widehat{c}_D(y) - \vec{c}_D(y)| = |(\widehat{c}_D - \vec{c}_D) \cdot \vec{y}| \leq \|\widehat{c}_D - \vec{c}_D\|_1 \leq \alpha'.$$

The first inequality follows from the fact that every monomial evaluates to 0 or 1 at the point y, and thus $\|y\|_\infty \leq 1$. This completes the proof of the theorem.

□

Using the second part of Lemma 5.4, we can improve the bound on the error at the expense of relaxing the privacy guarantee to (ε, δ)-differential privacy. This improved error only affects the hidden constants in Theorems 5.1-5.3, so we only state those theorems for ε-differential privacy.

Theorem 5.8. *Let $\mathcal{Q} = \{q_y\}_{y \in Y_\mathcal{Q} \subseteq \{0,1\}^m}$ be a set of counting queries over $\mathcal{X}$, and $\mathcal{F}_\mathcal{Q}$ be the $\mathcal{Q}$ function family (Definition 5.5). Assume that $\mathcal{S}$ efficiently and uniformly γ-approximates $\mathcal{F}_\mathcal{Q}$ on $Y_\mathcal{Q}$ with norm T (Definition 5.6). Then there is a sanitizer $\mathcal{M} \colon (\{0,1\}^d)^n \to \mathbb{R}^{|\mathcal{S}|}$ that*

1. *is (ε, δ)-differentially private,*
2. *runs in time $\mathrm{poly}(n, \log|\mathcal{X}|, |\mathcal{S}|, \log T, \log(1/\varepsilon), \log(1/\delta))$,*
3. *is (α, β)-accurate for $\mathcal{Q}$ for $\alpha = \gamma + \frac{12T|\mathcal{S}|^{3/2}\log(|\mathcal{S}|/\beta)\sqrt{\log(1/\delta)}}{\varepsilon n}$.*

The proof of this theorem is identical to that of Theorem 5.7, but using the second property of the Laplace mechanism from Theorem 5.4 in place of the first property.

5.4 Low-Dimensional Approximations

In this section we establish the existence of explicit low-dimensional linear approximations for the families $\mathcal{F}_\mathcal{Q}$ for some interesting query sets. All of our low-dimensional approximations are derived from *low-degree polynomials* with bounded coefficients.

An m-variate real multilinear polynomial $p_{\vec{c}} : \{0,1\}^m \to \mathbb{R}$ of *degree* t and *(L_∞-)norm* T can be written as

$$p_{\vec{c}}(y) = \sum_{S \subseteq [m]} c_S \prod_{\ell \in S} y_\ell$$

where $|c_S| \leq T$ for every $S \subseteq [m]$. Note that a polynomial of degree t is simply a linear combination of functions over the feature space $\mathcal{P}_t = \left\{p_S(y) = \prod_{\ell \in S} y_\ell \mid S \subseteq [m]\right\}$. If the polynomial has coefficients bounded in absolute value by T, the the linear combination of features in $\mathcal{P}_t$ has L_∞-norm bounded by T. The size of this feature space is $H_{m,t} := \sum_{j=0}^{t} \binom{m}{t}$. We remark that it is essentially without loss of generality to restrict to multilinear polynomials, since the domain is $\{0,1\}^m$.

Definition 5.9 (Uniform Approximation by Polynomials)**.** Given a family of m-variate functions $\mathcal{F} = \{f_x\}_{x \in \mathcal{X}}$ and a set $Y \subseteq \{0,1\}^m$, we say that the family $\mathcal{P}_t$ *uniformly γ-approximates $\mathcal{F}$ on* Y if for every $x \in \mathcal{X}$, there exists $\vec{c}_x \in [-T,T]^{H_{m,t}}$ such that

$$\forall y \in Y,\ |f_x(y) - p_{\vec{c}_x}(y)| \leq \gamma.$$

We say that $\mathcal{P}_t$ *efficiently and uniformly γ-approximates* $\mathcal{F}$ if there is an algorithm $\mathcal{P}_{\mathcal{F}}$ that takes $x \in \mathcal{X}$ as input, runs in time $\mathrm{poly}(\log|\mathcal{X}|, H_{m,t}, \log T)$, and outputs a coefficient vector $\vec{c}_x$ such that

$$\forall y \in Y,\ |f_x(y) - p_{\vec{c}_x}(y)| \leq \gamma.$$

5.4.1 Releasing Monotone Disjunctions

We define the class of monotone k-way disjunctions as follows:

Definition 5.10 (Monotone k-Way Disjunctions)**.** Let $\mathcal{X} = \{0,1\}^d$. The query set $\mathcal{Q}_{\mathrm{Disj},k} = \{q_y\}_{y \in Y_k \subseteq \{0,1\}^d}$ of *monotone k-way disjunctions over* $\{0,1\}^d$ contains a query q_y for every $y \in Y_k = \left\{y \in \{0,1\}^d \mid |y| \leq k\right\}$. Each query is defined as $q_y(x_1, \ldots, x_d) = \bigvee_{j=1}^{d} y_j x_j$. The $\mathcal{Q}_{\mathrm{Disj},k}$ function family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}} = \{f_x\}_{x \in \{0,1\}^d}$ contains a function $f_x(y_1, \ldots, y_d) = \bigvee_{j=1}^{d} y_j x_j$ for every $x \in \{0,1\}^d$.

Thus the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ consists of all disjunctions, and the image of $\mathcal{Q}_{\mathrm{Disj},k}$, which we denote Y_k, consists of all vectors $y \in \{0,1\}^d$ with at most k non-zero entries. We can approximate disjunctions over the set Y_k using a well-known transformation of the Chebyshev polynomials (see, e.g., [58] and [45]). First we recall the useful properties of the univariate Chebyshev polynomials.

Fact 5.11 (Chebyshev Polynomials)**.** *For every $k \in \mathbb{N}$ and $\gamma > 0$, there exists a univariate real polynomial $g_k(x) = \sum_{i=0}^{t_k} c_i x^i$ of degree t_k such that*

1. *$t_k = O(\sqrt{k}\log(1/\gamma))$,*
2. *for every $i \in \{0, 1, \ldots, t_k\}$, $|c_i| \leq 2^{O(\sqrt{k}\log(1/\gamma))}$,*
3. *$g_k(0) = 0$, and*
4. *for every $x \in \{1, \ldots, k\}$, $1 - \gamma \leq g_k(x) \leq 1 + \gamma$.*

Moreover, such a polynomial can be constructed in time $\mathrm{poly}(k, \log(1/\gamma))$ (e.g. using linear programming, though more efficient algorithms are known).

We can use Lemma 5.11 to approximate k-way monotone disjunctions. Note that our result easily extends to monotone k-way conjunctions via the identity $\wedge_{j=1}^{d} x_j y_j = 1 - \vee_{j=1}^{d}(1 - x_j)y_j$. Moreover, it extends to non-monotone conjunctions and disjunctions: we may extend the data universe as in [45, Theorem 1.2] to $\{0,1\}^{2d}$, and include the negation of each item in the original domain. Non-monotone conjunctions over domain $\{0,1\}^d$ correspond to monotone conjunctions over the expanded domain $\{0,1\}^{2d}$.

The next lemma shows that $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ can be efficiently and uniformly approximated by polynomials of low degree and low norm. The statement is a well-known application of Chebyshev polynomials, and a similar statement appears in [45] but without bounding the running time of the construction or a bound on the norm of the polynomials. We include the statement and a proof for completeness, and to verify the additional properties we need.

Lemma 5.12 (Approximating $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ by polynomials, similar to [45])**.** *For every $k, d \in \mathbb{N}$ such that $k \leq d$ and every $\gamma > 0$, the family $\mathcal{P}_t$ consisting of d-variate real polynomials of degree $t = O(\sqrt{k}\log(1/\gamma))$ and norm $T = d^{O(\sqrt{k}\log(1/\gamma))}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{Disj},k}}$ on the set Y_k.*

Proof. The algorithm $\mathcal{P}_{\mathrm{Disj},k}$ for constructing the polynomials appears in the relevant codebox above.

Since p_x is a degree-t_k polynomial applied to a degree-1 polynomial (in the variables y_j), its degree is at most t_k. To see the stated norm bound, note that every monomial of total degree

Input: a vector $x \in \{0,1\}^d$.

Let g_k be the polynomial described in Lemma 5.11.

Let $\vec{p}_x \in \mathbb{R}^{\binom{m+t_k}{t_k}}$ be the expansion of $p_x(y_1, \ldots, y_d) = g_k\left(\sum_{j=1}^d y_j x_j\right)$.

Output: $\vec{p}_x$.

Figure 14: $\mathcal{P}_{\text{Disj},k}$.

i in p_x comes from the expansion of $\left(\sum_{j=1}^d y_j x_j\right)^i$, and every coefficient in this expansion is a non-negative integer less than or equal to k^i. In p_x, each of these terms is multiplied by c_i (the i-th coefficient of g_k). Thus the norm of p_x is at most $\max_{i \in \{0,1,\ldots,t_k\}} k^i \cdot |c_i| = k^{O(\sqrt{k}\log(1/\gamma))} = d^{O(\sqrt{k}\log(1/\gamma))}$. To see that $\mathcal{P}_{\text{Disj},k}$ is efficient, note that we can find every coefficient of p_x of total degree i by expanding $\left(\sum_{j=1}^d y_j x_j\right)^i$ into all of its d^i terms and multiplying by c_i, which can be done in time $\text{poly}(d^{t_k}) = \text{poly}(\binom{d+t_k}{t_k})$, as is required.

To see that $\mathcal{P}_{\text{Disj},k}$ γ-approximates $\mathcal{F}_{\mathcal{Q}_{\text{Disj},k}}$, observe that for every $x, y \in \{0,1\}^d$, $f_x(y) = 0 \Rightarrow p_x(y) = 0$, and for every $x \in \{0,1\}^d, y \in Y_k$, $f_x(y) = 1 \Rightarrow 1-\gamma \leq p_x(y) \leq 1+\gamma$. This completes the proof. □

Theorem 5.1 in the introduction follows by combining Theorems 5.7 and 5.12.

5.4.2 Releasing Monotone r-of-k Queries

We define the class of monotone r-of-k queries as follows:

Definition 5.13 (Monotone r-of-k Queries)**.** Let $\mathcal{X} = \{0,1\}^d$ and $r, k \in \mathbb{N}$ such that $r \leq k \leq d$. The query set $\mathcal{Q}_{r,k} = \{q_y\}_{y \in Y_k \subseteq \{0,1\}^d}$ of *monotone r-of-k queries over* $\{0,1\}^d$ contains a query q_y for every $y \in Y_k = \left\{y \in \{0,1\}^d \mid |y| \leq k\right\}$. Each query is defined as $q_y(x_1, \ldots, x_d) = \mathbf{1}_{\sum_{j=1}^d y_j x_j \geq r}$. The $\mathcal{Q}_{r,k}$ function family $\mathcal{F}_{\mathcal{Q}_{r,k}} = \{f_x\}_{x \in \{0,1\}^d}$ contains a function $f_x(y_1, \ldots, y_d) = \mathbf{1}_{\sum_{j=1}^d y_j x_j \geq r}$ for every $x \in \{0,1\}^d$.

Sherstov [81, Lemma 3.11] gives an explicit construction of polynomials that can be used to approximate the family $\mathcal{F}_{\mathcal{Q}_{r,k}}$ over Y_k with low degree. It can be verified by inspecting the construction that the coefficients of the resulting polynomial are not too large.

Lemma 5.14 ([81])**.** *For every $r, k \in \mathbb{N}$ such that $r \leq k$ and $\gamma > 0$, there exists a univariate polynomial $g_{r,k} \colon \mathbb{R} \to \mathbb{R}$ of degree $t_{r,k}$ such that $g_{r,k}(x) = \sum_{i=0}^{t_k} c_i x^i$ and*

1. $t_{r,k} = O\left(\sqrt{rk}\log(k) + \sqrt{k\log(1/\gamma)\log(k)}\right)$,
2. *for every $i \in \{0, 1, \ldots, t_k\}$, $|c_i| \leq 2^{\widetilde{O}(\sqrt{kr\log(1/\gamma)})}$,*
3. *for every $x \in \{0, 1, \ldots, r-1\}$, $-\gamma \leq g_{r,k}(x) \leq \gamma$, and*
4. *for every $x \in \{r, \ldots, k\}$, $1 - \gamma \leq g_{r,k}(x) \leq 1 + \gamma$.*

Moreover, $g_{r,k}$ can be constructed in time $\mathrm{poly}(k, r, \log(1/\gamma))$ (e.g. using linear programming).

We can use these polynomials to approximate monotone r-of-k queries.

Lemma 5.15 (Approximating $\mathcal{F}_{\mathcal{Q}_{r,k}}$ on Y_k)**.** *For every $r, k, d \in \mathbb{N}$ such that $r \leq k \leq d$ and every $\gamma > 0$, the family $\mathcal{P}_t$ of d-variate real polynomials of degree $t = \widetilde{O}(\sqrt{kr\log(1/\gamma)})$ and norm $T = d^{\widetilde{O}(\sqrt{kr\log(1/\gamma)})}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{r,k}}$ on the set Y_k.*

Proof. The construction and proof is identical to that of Theorem 5.12 with the polynomials of Lemma 5.14 in place of the polynomials described in Lemma 5.11. □

Theorem 5.2 in the introduction now follows by combining Theorems 5.7 and 5.15. Note that our result also extends easily to non-monotone r-of-k queries in the same manner as Theorem 5.1.

Remark 5.16. Using the principle of inclusion-exclusion, the answer to a monotone r-of-k query can be written as a linear combination of the answers to $k^{O(r)}$ monotone k-way disjunctions. Thus, a sanitizer that is $(\alpha/k^{O(r)}, \beta)$-accurate for monotone k-way disjunctions implies a sanitizer that is (α, β)-accurate for monotone r-of-k queries. However, combining this implication with Theorem 5.1 yields a sanitizer with running time $d^{O(r\sqrt{k}\log(k/\beta))}$, which has a worse dependence on r than what we achieve in Theorem 5.2.

5.4.3 Releasing Decision Lists

A *length-k decision list* over m variables is a function which can be written in the form "if ℓ_1 then output b_1 else $\cdots$ else if ℓ_k then output b_k else output b_{k+1}," where each ℓ_i is a boolean literal in $\{x_1, \ldots, x_m\}$, and each b_i is an output bit in $\{0, 1\}$. Note that decision lists of length-k strictly

generalize k-way disjunctions and conjunctions. We use $\mathrm{DL}_{k,m}$ to denote the set of all length-k decision lists over m binary input variables.

Definition 5.17 (Evaluations of Length-k Decision Lists)**.** Let $k, m \in \mathbb{N}$ such that $k \leq m$ and $\mathcal{X} = \mathrm{DL}_{k,m}$. The query set $\mathcal{Q}_{\mathrm{DL}_{k,m}} = \{q_y\}_{y \in \{0,1\}^m}$ of *evaluations of length-k decision lists* contains a query q_y for every $y \in \{0,1\}^m$. Each query is defined as $q_y(x) = x(y)$ where $x \in \mathrm{DL}_{k,m}$ is a length-k decision list over m variables. The $\mathcal{Q}_{\mathrm{DL}_{k,m}}$ function family $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \{f_x\}_{x \in \mathrm{DL}_{k,m}}$ contains functions $f_x(y) = x(y)$ for every $x \in \mathrm{DL}_{k,m}$. That is, $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \mathrm{DL}_{k,m}$.

We clarify that in this setting, the records in the database are length-k decision lists over $\{0,1\}^m$ and the queries inputs in $\{0,1\}^m$. Thus $|\mathcal{X}| = |\mathrm{DL}_{k,m}| = m^{O(k)}$ and $|\mathcal{Q}| = 2^m$. Alternatively, $\mathcal{X} = \{0,1\}^d$ for $d = k(\log m + 2) + 1$, since a length-k decision list can be described using this many bits. Klivans and Servedio [58, Claim 5.4] have shown that decision lists of length k can be uniformly approximated to accuracy γ by low-degree polynomials.

Lemma 5.18 ([58])**.** *For every $k, m \in \mathbb{N}$ such that $k \leq m$ and every $\gamma > 0$, the family $\mathcal{P}_t$ of m-variate real polynomials of degree $\widetilde{O}\left(\sqrt{k}\log(1/\gamma)\right)$ and norm $T = m^{\widetilde{O}\left(\sqrt{k}\log(1/\gamma)\right)}$ efficiently and uniformly γ-approximates the family $\mathcal{F}_{\mathcal{Q}_{\mathrm{DL}_{k,m}}} = \mathrm{DL}_{k,m}$ on all of $\{0,1\}^m$.*

We obtain Theorem 5.3 of the introduction by combining Theorems 5.7 and 5.18.

Chapter 6

Faster Algorithms for Marginal Queries on Small Databases

6.1 Introduction

In Chapter 5, we presented a one-shot sanitizer for the set of k-way marginals that has both running time and minimum database size $d^{O(\sqrt{k})}$. Although this algorithm achieves the best-known running time for answering all k-way marginals, the minimum database size required is still significantly larger than $\widetilde{\Theta}(k\sqrt{d})$, which we know would be sufficient if we were not concerned with efficiency. In addition to the results of Chapter 5, there are several one-shot sanitizers whose running time is $\ll d^k$ and guarantee small average error for k-way marginals when the database size is $\ll d^k$ [41, 22, 45, 38]. However, even in the easiest case where we want small average error under the uniform distribution over all marginals, none of these algorithms matches private multiplicative weights with respect to the minimum database size (see Table 5.1)

However, recent experimental work of Hardt, Ligett, and McSherry [43] suggests that in some cases, it may be more important to optimize the minimum database than the running time. Indeed they show that for some databases of interest, even the 2^d-time private multiplicative weights algorithm is practical. They also demonstrate that more efficient algorithms based on adding independent noise do not provide good accuracy for these databases. These findings suggest that a promising approach to designing practical algorithms is to achieve a minimum database size comparable to that of private multiplicative weights (say, $\mathrm{poly}(d, k)$), and seek to optimize the running

time of the algorithm as much as possible under this constraint. Unfortunately, there is no algorithm we are aware of that has running time $2^{o(d)}$ and provides a meaningful accuracy guarantee for k-way marginals on a database of size comparable to $O(k\sqrt{d})$. If we want accurate answers to all k-way marginals then there is no $2^{o(d)}$ time algorithm with minimum database size $\mathrm{poly}(d, k)$.

In this chapter we give the first algorithms for privately answering marginal queries for this parameter regime. We show how to privately answer marginal queries in time $2^{o(d)}$ on databases of size $\widetilde{O}(kd^{.51})$, which is nearly the smallest a database can be while admitting any differentially private approximation to marginal queries.[13]

6.1.1 Our Results and Techniques

More specifically, we construct differentially private sanitizers (as opposed to one-shot santizers) for answering k-way marginal queries.

Theorem 6.1. *There exist constants $C_1, C_2 > 0$ such that for every $k, d, n \in \mathbb{N}$, $k \leq d$, and every $\varepsilon, \delta > 0$, there is an (ε, δ)-differentially private sanitizer that on input a database $D \in (\{0,1\}^d)^n$, with $n \geq C_2 \cdot d^{.51} \log |Q| \log(1/\delta)/\varepsilon$, runs in time*

$$\mathrm{poly}\left(n, |Q|, \min\left\{\exp\left(d^{1-1/C_1\sqrt{k}}\right), \exp\left(d/\log^{.99} d\right)\right\}\right)$$

and is $(.01, .01)$-accurate for the set of k-way marginal queries.

Theorem 6.1 easily gives rise to a one-shot sanitizer for k-way marginals. We can obtain this one-shot sanitizer simply by requesting answers to each of the $d^{\Theta(k)}$ distinct k-way marginal queries from the online sanitizer. In this case we obtain the following corollary.

Corollary 6.2. *There exist constants $C_1, C_2 > 0$ such that for every $k, d, n \in \mathbb{N}$, and every $\varepsilon, \delta > 0$, there is an (ε, δ)-differentially private one-shot sanitizer that, on input a database $D \in (\{0,1\}^d)^n$, with $n \geq C_2 \cdot kd^{.51} \log d \log(1/\delta)/\varepsilon$, runs in time*

$$\mathrm{poly}\left(n, \binom{d}{\leq k}, \min\left\{\exp\left(d^{1-1/C\sqrt{k}}\right), \exp\left(d/\log^{.99} d\right)\right\}\right)$$

and $(.01, .01)$-accurate for the set of k-way marginal queries.

[13] It is information-theoretically impossible for a differentially private algorithm to answer even 1-way marginal queries with non-trivial accuracy on a database of size $o(\sqrt{d}/\log d)$ [88, 90]

Table 2: A simplified summary of prior work on k-way marginals for $k \ll d$.

Paper	Running Time	Database Size
[26, 15, 30]	$d^{\Theta(k)}$	$O(d^{k/2})$
[16, 32, 35, 43]	$2^{O(d)}$	$\widetilde{O}(kd^{.5})$
Chapter 5	$d^{O(\sqrt{k})}$	$d^{O(\sqrt{k})}$
This chapter	$2^{O(d/\log^{.99} d)}$	$kd^{.5+o(1)}$
This chapter	$2^{d^{1-\Omega(1/\sqrt{k})}}$	$\widetilde{O}(kd^{.51})$

Here $\binom{d}{\leq k} := \sum_{j=0}^{k} \binom{d}{j}$, and the number of k-way marginal queries is polynomial in this quantity. Note that for $k = O(d/\log d)$, the per-query running time of the online sanitizer is the dominant term.

See Table 2 for a simplified summary of known one-shot sanitizers with error $\pm.01$ on every k-way marginal when $k \ll d$, including the results of this chapter. See Table 1 (Chapter 5) for a more complete summary including one-shot sanitizers with average case error. In Table 2, the running time statements ignore dependence on the database size and privacy parameters.

We make a few additional remarks about these results:

Remark 1. When $k = \Omega(\log^2 d)$, the minimum database size requirement can be improved to $|D| \geq Ckd^{.5+o(1)} \log(1/\delta)/\varepsilon$ (for some universal constant $C > 0$) without affecting the running time significantly, but we have stated the theorems with a looser bound for simplicity. Here the $o(1)$ term is a vanishing function of d.

Remark 2. Our algorithm can be modified so that instead of releasing approximate answers to each k-way marginal explicitly, it releases a summary of the database of size $\widetilde{O}(kd^{.01})$ from which an analyst can compute an approximate answer to any k-way marginal in time $\widetilde{O}(kd^{1.01})$.

Remark 3. Theorem 6.1 can actually be strengthened to give an *online sanitizer* for k-way marginal queries. Intuitively an online sanitizer is just a sanitizer where the input queries are given one at a time, and the sanitizer must answer each query immediately after it is issued. See [44] for a formal treatment of online sanitizers.

Recall that a monotone k-way disjunction is specified by a set $S \subseteq [d]$ of size k and asks what fraction of records in D have at least one of the attributes in S set to 1. Also recall that we view the

database as a function $f_D \colon \{0,1\}^d \to [0,1]$, in which each input vector $s \in \{0,1\}^d$ is interpreted as the indicator vector of a set $S \subseteq [d]$, and $f_D(s)$ equals the evaluation of the disjunction query specified by S on the database D. As in Chapter 5, our algorithm is based on techniques for approximating the function f_D.

The starting point for our algorithm is the online private multiplicative weights algorithm (PMW) [44], which has running time 2^d per query and answers any sequence of *arbitrary counting queries* provided $|D| \gtrsim \sqrt{d} \log |Q|$. Gupta, Roth, and Ullman [42] introduced the "IDC framework"—capturing PMW and other algorithms—for designing differentially private sanitizers and, in particular, showed that such an algorithm can be derived from any (non-privacy preserving) *online learning algorithm*.

Informally, an online learning algorithm is one that plays the following game: In each round, j, the learner receives a (possibly adaptively chosen) input s_j, and must produce a "guess" a_j as to the value of $f_D(s_j)$ for the unknown function f_D. After making each guess a_j, the learner is given some information about the value of $f_D(s_j)$. If the guess a_j is "far" from the true value $f_D(s_j)$, then we say that the learner has made a "mistake" in round j. Ultimately, for the sanitizer derived in the IDC framework, the notion of far will correspond to the accuracy, the per query running time will essentially be equal to the running time of the online learning algorithm, and the minimum database size required by the private algorithm will be proportional to the maximum number of mistakes that the learning algorithm makes in this game.

The algorithm of Hardt and Rothblum [44], is derived from an online learning algorithm that runs in time 2^d and makes $O(d)$ mistakes. A common approach to obtaining a faster online learning algorithm that still makes few mistakes is to use a *polynomial approximation*[14] to the target function f_D. Indeed, it is well-known in computational learning theory that if f_D can be approximated to within high accuracy on every input by a d-variate polynomial $p_D \colon \{0,1\}^d \to \mathbb{R}$ of degree t and weight at most W (where the weight is defined to be the sum of the absolute values of the coefficients), then there is an online learning algorithm that runs in time $\mathrm{poly}\left(\binom{d}{\leq t}\right)$ and makes $O(Wd)$ mistakes. Thus, if $t \ll d$, the running time of such an online learning algorithm will be significantly less than 2^d and the number of mistakes (and thus the minimum database size of the

[14]For this informal introduction, we are using the term polynomial a bit loosely. More precisely, we will approximate $f_D : \{0,1\}^m \to [0,1]$ by a low-weight linear combination of low-width parity functions over $\{0,1\}^m$, which could alternatively be viewed as a polynomial approximation if wrote f_D as a function mapping $\{-1,1\}^m$ to $[-1,1]$ in the standard fashion.

resulting private algorithm) will only increase by a factor of W.

In order to obtain a faster online learning algorithm that makes few mistakes, it suffices to show that for every database, D, there is a low-degree, low-weight polynomial p_D such that $|p_D(s) - f_D(s)|$ is small for all vectors $s \in \{0,1\}^d$ corresponding to monotone k-way disjunction queries. As we implicitly showed in Chapter 5, it is sufficient to construct a low-degree, low-weight polynomial that can approximate the d-variate OR function on inputs $s \in \{0,1\}^d$ of Hamming weight at most k. Indeed, we show that for every $k \leq d$, there is a suitable polynomial of degree $d^{1-\Omega(1/\sqrt{k})}$ and weight $d^{.01}$. For larger values of k, this upper bound approaches the trivial upper bound of d. However, we also show that for the hardest case of $k = d$, there exists a suitable polynomial of degree $d/\log^{.99} d$ and weight $d^{o(1)}$.

If our construction could be improved to give a polynomial of even smaller degree and weight $\mathrm{poly}(d)$ that approximates the OR function on all inputs of Hamming weight at most k, then it would immediately imply a faster algorithm for answering k-way marginals with a similar accuracy guarantee. However, we prove a lower bound showing that it is not possible to significantly improve on our construction. We believe this lower bound is of independent approximation-theoretic interest.

Theorem 6.3. *Let $OR_d : \{0,1\}^d \to \{0,1\}$ denote the OR function on d variables, and for every vector $x \in \{0,1\}^d$, let $|x|$ denote $\sum_{i=1}^d x_i$, the Hamming weight of x. Fix $k = o(\log d)$, and let p be a real d-variate polynomial satisfying $|p(x) - OR_d(x)| \leq 1/6$ for all $x \in \{0,1\}^d$ with $|x| \leq k$. If the sum of the absolute values of the coefficients of p is bounded by $d^{O(1)}$, then the total degree of p is at least $d^{1-O(1/\sqrt{k})}$.*

As was the case in Chapter 5, our algorithm can be extended to the case where f_D is approximated not by a low-degree polynomial, but rather by a low-weight linear combination of functions from an arbitrary feature space $\mathcal{S}$. In this case the running time will be proportional to $|\mathcal{S}|$. Theorem 6.3 shows that we cannot construct a smaller feature space containing a low-weight approximation to f_D by restricting to monomials of lower degree. We leave it as an interesting open question to determine whether or not there exists a smaller feature space such that every disjunction over d variables can be approximated on inputs of low Hamming weight by a low-weight linear combination of these features. A positive answer to this question would immediately yield a more efficient algorithm for answering k-way marginals with a similar accuracy guarantee to ours.

6.2 Preliminaries

6.2.1 Query Function Families

We recall the notion of a query-function family from Chapter 5.

Definition 6.4 ($\mathcal{Q}$-Function Family)**.** Let $\mathcal{Q} = \{q_y\}_{y \in Y_{\mathcal{Q}} \subseteq \{0,1\}^m}$ be a set of counting queries on a data universe $\mathcal{X}$, where each query is indexed by an m-bit string. We define the *index set of* $\mathcal{Q}$ to be the set $Y_{\mathcal{Q}} = \{y \in \{0,1\}^m \mid q_y \in \mathcal{Q}\}$.

We define the *$\mathcal{Q}$-function family* $\mathcal{F}_{\mathcal{Q}} = \{f_x : \{0,1\}^m \to \{0,1\}\}_{x \in \mathcal{X}}$ as follows: For every possible database row $x \in \mathcal{X}$, the function $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ is defined as $f_{\mathcal{Q},x}(y) = q_y(x)$. Given a database $D \in \mathcal{X}^n$ we define the function $f_{\mathcal{Q},D} : \{0,1\}^m \to [0,1]$ where $f_{\mathcal{Q},D}(q) = \frac{1}{n}\sum_{i=1}^{n} f_{\mathcal{Q},x^{(i)}}(q)$. When $\mathcal{Q}$ is clear from context we will drop the subscript $\mathcal{Q}$ and simply write f_x, f_D, and $\mathcal{F}$.

For some intuition about this transformation, when the queries are monotone k-way disjunctions on a database $D \in (\{0,1\}^d)^n$, the queries are defined by sets $S \subseteq [d]$, $|S| \leq k$. In this case each query can be represented by the d-bit indicator vector of the set S, with at most k non-zero entries. Thus we can take $m = d$ and $Y_{\mathcal{Q}} = \left\{y \in \{0,1\}^d \mid \sum_{j=1}^{d} y_j \leq k\right\}$.

6.2.2 Low-Weight Linear Approximations

As in Section 5.2.3, let $\mathcal{S} = \{s \colon \{0,1\}^m \to [-1,1]\}$ be a *feature space* of efficiently (time $\mathrm{poly}(m)$) computable m-variate functions. For a vector $\vec{c} = (c_s)_{s \in \mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$, let $g_{\vec{c}}$ be the function

$$g_{\vec{c}}(y) = \sum_{s \in \mathcal{S}} c_s \cdot s(y).$$

Abusing notation, we will often associate the function $g_{\vec{c}}$ with its coefficient vector $\vec{c}$. Notice that computing $g_{\vec{c}}(y)$ is equivalent to computing $\langle \vec{c}, \vec{y} \rangle$, where $\vec{y} \in [0,1]^{|\mathcal{S}|}$ is the vector $(y_s)_{s \in \mathcal{S}}$ formed by taking $\vec{y}_s = s(y)$ for every $s \in \mathcal{S}$. In contrast to Section 5.2.3, we now wish to work with linear combinations of function in $\mathcal{S}$ that have bounded L_1*-norm* (as opposed to L_∞-norm). To reduce ambiguity between different norms, we define the *weight of* $g_{\vec{c}}$ to be

$$W(g_{\vec{c}}) = \|\vec{c}\|_1 = \sum_{s \in \mathcal{S}} |\vec{c}_s|.$$

We will tend to restrict attention to functions $g_{\vec{c}}$ that have weight *exactly* W. This restriction is essentially without loss of generality if we assume that $\mathcal{S}$ includes a function that is identically 0. Then if we are given a function $g_{\vec{c}}$ over $\mathcal{S}$ with weight $W' < W$, we can place additional weight of $W - W'$ on the identically 0 function so that $g_{\vec{c}}$ has weight exactly W. We also restrict attention to functions $g_{\vec{c}}$ such that $\vec{c}$ lies in $\mathbb{R}_{\geq 0}^{|\mathcal{S}|}$. This restriction is also essentially without loss of generality if we assume that $\mathcal{S}$ is closed under negation ($s \in \mathcal{S} \Rightarrow -s \in \mathcal{S}$). If either of these two restrictions fails to hold for a given feature space $\mathcal{S}$, then we can obtain a new feature space of size $2|\mathcal{S}| + 1$ for which they do hold, which will not have a significant effect on the statement of our results.

We are interested in cases where the function $f_{\mathcal{Q},x} : \{0,1\}^m \to \{0,1\}$ can be approximated well on all the indices in $Y_{\mathcal{Q}}$ by a linear combination of functions in $\mathcal{S}$ with low weight. Formally:

Definition 6.5 (Uniform Approximation by a Feature Space)**.** Given a family of m-variate functions $\mathcal{F} = \{f_x \colon \{0,1\}^m \to [-1,1]\}_{x \in \mathcal{X}}$ and a set $Y \subseteq \{0,1\}^m$, we say that the feature space $\mathcal{S} = \{s \colon \{0,1\}^m \to [-1,1]\}$ *uniformly γ-approximates $\mathcal{F}$ on Y with weight W* if for every $x \in \mathcal{X}$, there exists $\vec{c}_x \in \mathbb{R}_{\geq 0}^{|\mathcal{S}|}$ such that $W(\vec{c}) = W$ and

$$\forall y \in Y,\ |f_x(y) - \langle \vec{c}_x, \vec{y} \rangle| \leq \gamma.$$

6.3 From Low-Weight Approximations to Sanitizers

In this section we show that low-weight polynomial approximations imply data release algorithms that provide approximate answers even on small databases. Informally, if a feature space $\mathcal{S}$ of m-variate functions γ-approximates $\mathcal{F}$ with weight W, for a sufficiently small constant γ, then there is a differentially private online algorithm with running time $\mathrm{poly}(m, |\mathcal{S}|)$ that releases answers to every sequence of queries Q in $\mathcal{Q}$ within error $\pm.01$ as long as $n \gtrsim W\sqrt{m} \log |Q|/\varepsilon$.

We will prove this result by constructing a new IDC that has a good mistake bound and running time for families of queries that can be approximated with low-weight over a small feature space. We then invoke Theorem 2.15 from Section 2.4.3 to obtain our differentially private sanitizer.

6.3.1 From Low-Weight Approximations to IDCs

The new IDC is derived from the standard (non-privacy-preserving) multiplicative weights algorithm, and is specified in Figure 15. For this IDC, the data structure approximating the database

will be a low-weight linear combination of features from a feature space $\mathcal{S}$, which is a coefficient vector $\vec{c}^{(t)} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}|}$.

For a given vector $\vec{c} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}|}$ of weight W, we define $\overline{c} = \vec{c}/W$. We observe two things about $\overline{c}$: (1) Given a query $y \in \{0,1\}^m$, we can construct a vector $\vec{y}$ of length $|\mathcal{S}|$ in which $\vec{y}_s = s(y_i)$ for every $s \in \mathcal{S}$, and this vector will satisfy $W\langle \overline{c}, \vec{y} \rangle = \langle \vec{c}, \vec{y} \rangle$. (2) $\overline{c}$ now represents a probability distribution on the $|\mathcal{S}|$ coefficients.

$\mathbf{U}^{MW}_{\alpha,\mathcal{S},W}(\vec{c}^{(t)}, y^{(t)}, \widehat{a}^{(t)})$:

Input: An approximation $\vec{c}^{(t)}$ over feature space $\mathcal{S}$, a query $y^{(t)} \in \mathcal{Q}$, and $\widehat{a}^{(t)} \in \mathbb{R}$.

If: $\vec{c}^{(t)} = \emptyset$ **then: output** $\vec{c}^{(t+1)} = (1, \ldots, 1)$ and **halt.**

Let $\overline{c}^{(t)} = \vec{c}^{(t)}/W$ and let $\eta \leftarrow \alpha/4W$.

If: $\widehat{a}^{(t)} < W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle$

Let $\overline{r}^{(t)} = \vec{y}^{(t)}$

Else:

Let $\overline{r}^{(t)} = -\vec{y}^{(t)}$

Update: For all $s \in \mathcal{S}$ let

$$\overline{c}_s^{(t+1)} \leftarrow \exp(-\eta \overline{r}_s^{(t)}) \cdot \overline{c}_s^{(t)}$$

$$\overline{c}^{(t+1)} \leftarrow \frac{\overline{c}^{(t+1)}}{\|\overline{c}^{(t+1)}\|_1}$$

Output $\vec{c}^{(t+1)} = W \cdot \overline{c}^{(t+1)}$.

Figure 15: The Multiplicative Weights Algorithm for Low-Weight Feature Spaces.

We summarize the properties of the multiplicative weights algorithm in the following theorem:

Theorem 6.6. *For every $\alpha > 0$, and every family of counting queries $\mathcal{Q}$ if $\mathcal{S}$ $(\alpha/4)$-uniformly approximates $\mathcal{F}$ on Y with weight W then $\mathbf{U}^{MW}$ (Figure 15) is an iterative database construction with mistake bound B for $\mathcal{Q}$for*

$$B = \frac{16W^2 \log |\mathcal{S}|}{\alpha^2}.$$

Moreover, $\mathbf{U}$ runs in time $\mathrm{poly}\big(|\mathcal{S}|, \log(1/\alpha), \log W\big)$.

Proof. Let $D \in (\{0,1\}^d)^n$ be any database and consider a $(\mathbf{U}^{MW}, D, \mathcal{Q}, \alpha, B)$-database update sequence, $\big\{(\vec{c}^{(t)}, y^{(t)}, \widehat{a}^{(t)})\big\}_{t=1,\ldots,B}$. It will be sufficient if we can show that $B \leq 16W^2 \log |\mathcal{S}|/\alpha^2$.

Specifically, that after $B = 16W^2 \log|\mathcal{S}|/\alpha^2$ invocations of $\mathbf{U}^{MW}$, the function $\vec{c}^{(B)}$ is such that

$$\forall y \in Y_{\mathcal{Q}},\ |\langle \vec{c}^{(B)}, \vec{y}\rangle - f_D(y)| \leq \alpha.$$

That is, $\vec{c}^{(B)}$ represents a function that approximates f_D.

The assumption of our theorem is that for every $x^{(i)} \in D$, there exists $\vec{c}_{x^{(i)}} \in \mathbb{R}^{|\mathcal{S}|}_{\geq 0}$, of weight W, such that

$$\forall y \in Y_{\mathcal{Q}},\ |\langle \vec{c}_{x^{(i)}}, \vec{y}\rangle - f_{x^{(i)}}(y)| \leq \frac{\alpha}{4}.$$

Thus, since $f_D = \frac{1}{n}\sum_{i=1}^{n} f_{x^{(i)}}$, the function $\vec{c}_D = \frac{1}{n}\sum_{i=1}^{n} \vec{c}_{x^{(i)}}$ will satisfy

$$\forall y \in Y_{\mathcal{Q}},\ |\langle \vec{c}_D, \vec{y}\rangle - f_D(y)| \leq \frac{\alpha}{4}.$$

Note that $\vec{c}_D \in \mathbb{R}^{|\mathcal{S}|}_{\geq 0}$, thus, viewing $\vec{c}_D$ as a vector, and considering its normalized version $\overline{c}_D$, we have

$$\forall y \in Y_{\mathcal{Q}},\ |W\langle \overline{c}_D, \vec{y}\rangle - f_D(y)| \leq \frac{\alpha}{4}.$$

Given the existence of $\overline{c}_D$, we will define a potential function capturing how far $\overline{c}^{(t)}$ is from $\overline{c}_D$. Specifically, we define

$$\Psi_t := KL(\overline{c}_D || \overline{c}^{(t)}) = \sum_{s \in \mathcal{S}} \overline{c}_{D,s} \log\left(\frac{\overline{c}_{D,s}}{\overline{c}^{(t)}_s}\right)$$

to be the KL divergence between $\overline{c}_D$ and the current approximation $\overline{c}^{(t)}$. We have the following fact about KL divergence.

Fact 6.7. *For all t: $\Psi_t \geq 0$, and $\Psi_0 \leq \log|\mathcal{S}|$.*

We will argue that in each step the potential drops by at least $\alpha^2/16W^2$. Because the potential begins at $\log|\mathcal{S}|$, and must always be non-negative, we know that there can be at most $B \leq 16W^2 \log|\mathcal{S}|/\alpha^2$ steps before the algorithm outputs a (vector representation of) a low-weight linear combination over $\mathcal{S}$ that approximates f_D on $Y_{\mathcal{Q}}$.

The following lemma is standard in the analysis of multiplicative-weights-based algorithms. We include a proof for completeness.

Lemma 6.8 (See e.g. [5])**.**

$$\Psi_t - \Psi_{t+1} \geq \eta\left(\langle \overline{c}^{(t)}, r^{(t)}\rangle - \langle \overline{c}_D, r^{(t)}\rangle\right) - \eta^2$$

Proof. We can prove the lemma via the following calculation:

$$\begin{aligned}
\Psi_t - \Psi_{t+1} &= \sum_{s \in \mathcal{S}} \overline{c}_{D,s} \log\left(\frac{\overline{c}_{D,s}}{\overline{c}_s^{(t)}}\right) - \sum_{s \in \mathcal{S}} \overline{c}_{D,s} \log\left(\frac{\overline{c}_{D,s}}{\overline{c}_s^{(t+1)}}\right) \\
&= \sum_{s \in \mathcal{S}} \overline{c}_{D,s} \log\left(\frac{\overline{c}_s^{(t+1)}}{\overline{c}_s^{(t)}}\right) \\
&= \sum_{s \in \mathcal{S}} \overline{c}_{D,s} \log\left(\frac{\exp(-\eta r_s^{(t)})}{\sum_{u \in \mathcal{S}} \exp(-\eta r_u^{(t)}) \overline{c}_u^{(t)}}\right) \\
&= -\eta \langle \overline{c}_D, r^{(t)} \rangle - \log\left(\sum_{u \in \mathcal{S}} \exp(-\eta r_u^{(t)}) \overline{c}_u^{(t)}\right) \\
&\geq -\eta \langle \overline{c}_D, r^{(t)} \rangle - \log\left(\sum_{u \in \mathcal{S}} \overline{c}_u^{(t)} (1 + (\eta r_u^{(t)})^2 - \eta r_u^{(t)})\right) \\
&\geq -\eta \langle \overline{c}_D, r^{(t)} \rangle - \log\left(\sum_{u \in \mathcal{S}} \overline{c}_u^{(t)} ((\eta r_u^{(t)})^2 - \eta r_u^{(t)})\right) \\
&\geq -\eta \langle \overline{c}_D, r^{(t)} \rangle - \sum_{u \in \mathcal{S}} \overline{c}_u^{(t)} ((\eta r_u^{(t)})^2 - \eta r_u^{(t)}) \\
&= \eta \left(\langle \overline{c}^{(t)}, r^{(t)} \rangle - \langle \overline{c}_D, r^{(t)} \rangle \right) - \eta^2
\end{aligned}$$

□

In this calculation, we have used the fact that, by construction, $r_u^{(t)} \in [0, 1]$ and using the elementary facts that for $x > 0$, $\log(1 + x) \leq x$ and $e^{-x} \leq 1 - x + x^2$.

The rest of the proof now follows easily. By the conditions of an iterative database construction algorithm, $|\widehat{a}^{(t)} - f_D(y^{(t)})| \leq \alpha/2$. Hence, for each t such that $|W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle - f_D(y^{(t)})| \geq \alpha$, we also have that $W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle > f_D(y^{(t)})$ if and only if $W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle > \widehat{a}^{(t)}$.

In particular, if $r^{(t)} = y^{(t)}$, then $W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle - W\langle \overline{c}_D, \vec{y}^{(t)} \rangle \geq \alpha/2$. Similarly, if $r^{(t)} = -y^{(t)}$, then $W\langle \overline{c}_D, \vec{y}^{(t)} \rangle - W\langle \overline{c}^{(t)}, \vec{y}^{(t)} \rangle \geq \alpha$. Here we have utilized the fact that $|\vec{c}_D(y) - f_D(y)| \leq \alpha/4$. Therefore, by Lemma 6.8 and the fact that $\eta = \alpha/4W$:

$$\Psi_t - \Psi_{t+1} \geq \frac{\alpha}{4W} \left(\langle \overline{c}^{(t)}, r^{(t)} \rangle - \langle \overline{c}_D, r^{(t)} \rangle \right) - \frac{\alpha^2}{16W^2} \geq \frac{\alpha}{4W} \left(\frac{\alpha}{2W} \right) - \frac{\alpha^2}{16W^2} = \frac{\alpha^2}{16W^2}$$

□

6.4 Low-Weight Approximations

In the previous section we reduced the problem of designing efficient online algorithms for marginal queries to designing certain low-weight linear approximations. In this section we show that such low-weight linear approximations exist using low-degree, low-weight polynomials. In this and the following section we give upper and lower bounds on the degree of suitable low-weight polynomials. For technical convenience, and consistency with the literature in approximation theory, the notation used for these sections differs somewhat from what we have used in Chapter 5 and in the previous sections. This section and the next are intended to be self-contained, and in Theorem 6.10 we clarify the relationship between the two settings.

In this section we are interested in m-variate real polynomials $p\colon \{-1,1\}^m \to \mathbb{R}$, written as

$$p(y) = \sum_{S\subseteq[m]} c_S \cdot \prod_{i\in S} y_i.$$

We define the *degree* of the polynomial to be

$$\deg(p) := \max\{|S| : S \subseteq [m], c_S \neq 0\},$$

and we define the *weight* $w(.)$ and *non-constant weight* $w^*(.)$ of the polynomial to be

$$w(p) := \sum_{S\subseteq[m]} |c_S|, \qquad w^*(p) := \sum_{S\subseteq[m],S\neq\emptyset} |c_S|.$$

We use $\binom{[m]}{\leq t}$ to denote $\{S \subseteq [m] \mid |S| \leq t\}$ and $\binom{m}{\leq t} = \left|\binom{[m]}{\leq t}\right| = \sum_{j=0}^{t}\binom{m}{j}$.

The next definition is similar to Definition 6.5 but with different notation and specialized to polynomials.

Definition 6.9 (Restricted Approximation by Polynomials)**.** Given a function $f\colon Y \to \mathbb{R}$, where $Y \subseteq \mathbb{R}^m$, and a subset $Y' \subseteq Y$, we denote the restriction of f to Y' by $f|_{Y'}$. Given an m-variate real polynomial p, we say that p is a γ-*approximation* to the restriction $f|_{Y'}$, if $|f(y) - p(y)| \leq \gamma$ $\forall y \in Y'$. Notice there is no restriction whatsoever placed on $p(y)$ for $y \in Y \setminus Y'$.

Given a family of m-variate functions $\mathcal{F} = \{f_x\colon Y \to \mathbb{R}\}_{x\in\mathcal{X}}$, where $Y \subseteq \mathbb{R}^m$, a set $Y' \subseteq Y$ and a family $\mathcal{P}$ of m-variate real polynomials, we say that the family $\mathcal{P}$ is a γ-*approximation to* $\mathcal{F}|_{Y'}$ if for every $x \in \mathcal{X}$, there exists $p_x \in \mathcal{P}$ that is a γ-approximation to $f_x|_{Y'}$.

We view the d variate OR function, OR_d as mapping inputs from $\{-1,1\}^d$ to $\{-1,1\}$, with the convention that -1 is TRUE and 1 is FALSE. Let $H_{m,k} = \{x \in \{-1,1\}^m : \sum_{i=1}^m (1-x_i)/2 \leq k\}$ denote the set of inputs of Hamming weight at most k. Let $\mathcal{P}_{t,W}$ denote the family of all m-variate real polynomials of degree t and weight W. For the upper bound, we will show that for certain small values of t and W, the family $\mathcal{P}_{t,W}$ is a γ-approximation to the family of all disjunctions restricted to $H_{m,k}$.

The next theorem reconciles the notation and terminology of this section with that of the previous sections.

Theorem 6.10. *Let $\mathcal{Q}$ be the set of all monotone k-way disjunctions on $\{0,1\}^d$ and let $\mathcal{F}_\mathcal{Q}$ and $Y_\mathcal{Q}$ be the corresponding function family and index set. Here, a k-way disjunction $q_y(x) = \bigvee_{i:y_i=1} x_i$ is specified by its indicator vector $y \in Y_\mathcal{Q} = \left\{y \in \{0,1\}^d \mid \sum_{j=1}^d y_j \leq k\right\}$. Let $\mathcal{F}_\mathcal{Q}$ be the corresponding function family. Assume that there is a d-variate polynomial of degree at most t and weight at most W that γ-approximates the function OR_d on all inputs in $H_{m,k}$. Then there exists a feature space of size at most $2\binom{d}{\leq t} + 1$ that γ-uniformly approximates the family $\mathcal{F}$ on Y with weight W.*

Proof of Theorem 6.10. For any $x \in \{0,1\}^d$, let $OR_{d,x}(y) = \bigvee_{i|x_i=1} y_i$. We note that $q_y(x) = OR_{d,x}(y)$ Suppose that there is a polynomial p such that

$$\forall y \in H_{m,k},\ |p(y) - OR_d(y)| \leq \gamma.$$

Then the polynomial $p_x(y) = p(y_1^{x_1}, \ldots, y_d^{x_d})$ is such that

$$\forall y \in H_{m,k},\ |p_x(y) - OR_{d,x}(y)| \leq \gamma.$$

Further, it is easy to see that (since $x_i \in \{0,1\}$), the degree and weight of p_x are no larger than that of p. Now observe that the functions $f_x \in \mathcal{F}$ are exactly the functions $OR_{d,x}$, except that f_x maps $\{0,1\}^d \to \{0,1\}$ (with 1 representing logical TRUE), whereas $OR_{d,x}$ maps $\{-1,1\}^d \to \{-1,1\}$ (with -1 representing logical TRUE). Thus, for every monomial $\prod_{i\in S} y_i$, we can instead use the feature $\frac{1}{2} - \frac{1}{2}\prod_{i\in S}(1-2y_i)$ to obtain an equivalent function mapping $\{0,1\}^d \to \{0,1\}$. Thus, the feature space $\mathcal{S}_t = \left\{s_S(y) = \frac{1}{2} - \frac{1}{2}\prod_{i\in S}(1-2y_i) \mid S \subseteq [d], |S| \leq t\right\}$, will γ-approximate the family $\mathcal{F}$ on Y with weight at most W. Also note that the size of this feature space is $\binom{d}{\leq t}$. Finally, and we will expand the feature set to be closed under negation and contain a constant 0 feature

to be consistent with the requirement in Definition 6.5 that the weight of the approximation be exactly W and that the coefficients be non-negative. This expanded feature space will have size $2\binom{d}{\leq t} + 1$. □

With Theorem 6.10 in hand, we can focus on understanding the choices of weight and degree for which there admits a polynomial approximation to the function OR_d.

For our lower bound, we will show that any collection of polynomials with small weight that is a γ-approximation to the family of disjunctions restricted to $H_{m,k}$ should have large degree. We need the following definitions in place:

Definition 6.11 (Approximate Degree)**.** Given a function $f\colon Y \to \mathbb{R}$, where $Y \subseteq \mathbb{R}^m$, the γ-*approximate degree* of f is

$$\deg_\gamma(f) := \min\{d : \exists \text{ real polynomial } p \text{ that is a } \gamma\text{-approximation to } f, \deg(p) = d\}.$$

Analogously, the (γ, W)-*approximate degree* of f is

$$\deg_{(\gamma,W)}(f) := \min\{d : \exists \text{polynomial } p \text{ that is a } \gamma\text{-approximation to } f, \deg(p) = d, w(p) \leq W\}.$$

It is clear that $\deg_\gamma(f) = \deg_{(\gamma,\infty)}(f)$.

We let $w^*(f,t)$ denote the *degree-t non-constant margin weight* of f, defined to be:

$$w^*(f,t) := \min\{w^*(p) : \deg(p) \leq t, f(y)p(y) \geq 1 \; \forall \; y \in Y\}.$$

The above definitions extend naturally to the restricted function $f|_{Y'}$.

Our definition of non-constant margin weight is closely related to the well-studied notion of the degree-t polynomial threshold function (PTF) weight of f (see e.g. [83]), which is defined as $\min_p w(p)$, where the minimum is taken over all degree-t polynomials p with integer coefficients, such that $f(x) = \text{sign}(p(x))$ for all $x \in \{-1,1\}^d$. Often, when studying PTF weight, the requirement that p have integer coefficients is used only to ensure that p has non-trivial margin, i.e. that $|p(x)| \geq 1$ for all $x \in \{-1,1\}^d$; this is precisely the requirement captured in our definition of non-constant margin weight. We choose to work with margin weight because it is a cleaner quantity to analyze using linear programming duality; PTF weight can also be studied using LP duality, but the integrality constraints on the coefficients of p introduces an integrality gap that causes some loss in the analysis (see e.g. Sherstov [83] and Klauck [57]).

The OR_d function is easily seen to have an exact polynomial representation of constant weight and degree d (e.g. see Fact 6.14 below); however, an approximation with smaller degree may be achieved at the expense of larger weight. The best known weight-degree tradeoff (implicit in Servedio et al. [80]), can be stated as follows: there exists a polynomial p of degree t and weight $(d \log(1/\gamma)/t)^{(d(\log 1/\gamma)^2/t)}$ that γ-approximates the OR_d function on all Boolean inputs, for every t larger than $\sqrt{d} \log(1/\gamma)$. Setting the degree t to be $O(d/\log^{.99} d)$ yields a polynomial of weight at most $d^{0.01}$ that approximates the OR_d function over the entire Boolean hypercube to any desired constant accuracy. On the other hand, Lemma 8 of [80] can be shown to imply that any polynomial of weight W that $1/3$-approximates the OR_d function requires degree $\Omega(d/\log W)$, essentially matching the $O(d/\log^{.99} d)$ upper bound of Servedio et al. when $W = d^{O(1)}$ in Lemma 6.12.

However, in order to privately release k-way marginals, we have shown that it suffices to construct polynomials that are accurate *only* on inputs of low Hamming weight. In this section, we give a construction that achieves significantly improved weight degree trade-offs in this setting. In the next section, we demonstrate the tightness of our construction by proving matching lower bounds.

We construct our approximations by decomposing the d-variate OR function into an OR of OR's, which is the same approach taken by Servedio et al. [80]. Here, the outer OR has fan-in m and the inner OR has fan-in d/m, where the subsequent analysis will determine the appropriate choice of m. In order to obtain an approximation that is accurate on all Boolean inputs, Servedio et al. approximate the outer OR using a transformation of the Chebyshev polynomials of degree $\sqrt{m}$, and compute each of the inner OR's exactly.

For $k \ll \log^2 d$, we are able to substantially reduce the degree of the approximating polynomial, relative to the construction of Servedio et al., by leveraging the fact that we are interested in approximations that are only accurate on inputs of Hamming weight at most k. Specifically, we are able to approximate the outer OR function using a polynomial of degree only $\sqrt{k}$ rather than $\sqrt{m}$, and argue that the weight of the resulting polynomial is still bounded by a polynomial in d.

We now proceed to prove the main lemmas. For the sake of intuition, we begin with weight-degree tradeoffs in the simpler setting in which we are concerned with approximating the OR_d function over the entire Boolean hypercube. The following lemma, proved below for completeness, is implicit in the work of [80].

Lemma 6.12. *For every $\gamma > 0$ and $m \in [d]$, there is a real-valued polynomial of degree $t =$*

$O(d\log(1/\gamma)/\sqrt{m})$ *and weight* $W = m^{O(\sqrt{m}\log(1/\gamma)}$ *that* γ*-approximates the* OR_d *function.*

Our main contribution in this section is the following lemma that gives an improved polynomial approximation to the OR_d function restricted to inputs of low Hamming weight.

Lemma 6.13. *For every* $\gamma > 0$*,* $k < d$ *and integer* $k < m \leq d$*, there is a polynomial of degree* $t = O(d\sqrt{k}\log(1/\gamma)/m)$ *and weight* $W = m^{O(\sqrt{k}\log(1/\gamma))}$ *that* γ*-approximates the* OR_d *function restricted to inputs of Hamming weight at most* k*.*

For any constant accuracy γ, one may take $m = d^{O(1/\sqrt{k})}$ in the lemma (here the choice of constant depends on the constants in Fact 6.15 and the desired accuracy) and obtain a polynomial of degree $d^{1-\Omega(1/\sqrt{k})}$ and weight $d^{.01}$.

Our constructions use the following basic facts.

Fact 6.14. *The real polynomial* $p_d : \{-1,1\}^d \to \mathbb{R}$

$$p_d(x) = 2\left(\sum_{S\subseteq[d]} 2^{-d}\prod_{i\in S} x_i\right) - 1 = 2\prod_{i=1}^{d}\left(\frac{1+x_i}{2}\right) - 1$$

computes $\mathrm{OR}_d(x)$ *and has weight* $w(p_d) \leq 3$*.*

Fact 6.15. *For every* $k \in \mathbb{N}$ *and* $\gamma > 0$*, there exists a univariate real-valued polynomial* $p = \sum_{i=0}^{t_k} c_i x^i$ *of degree* t_k *such that*

1. $t_k = O(\sqrt{k}\log(1/\gamma))$*,*
2. *for every* $i \in \{1,\ldots,t_k\}$*,* $|c_i| \leq 2^{O(\sqrt{k}\log(1/\gamma))}$*,*
3. $p(0) = 0$*, and*
4. *for every* $x \in \{1,\ldots,2k\}$*,* $|p(x) - 1| \leq \gamma/2$*.*

Proof of Lemma 6.12. We can compute $\mathrm{OR}_d(y)$ as a disjunction of disjunctions by partitioning the inputs $y_1,\ldots,y_d$ into blocks of size d/m and computing:

$$\mathrm{OR}_m(\mathrm{OR}_{d/m}(y_1,\ldots,y_{d/m}),\ldots,\mathrm{OR}_{d/m}(y_{d-d/m+1},\ldots,y_d)).$$

In order to approximately compute $\mathrm{OR}_d(y)$, we compute the inner disjunctions exactly using the polynomial $p_{d/m}$ given in Fact 6.14 and approximate the outer disjunction using the polynomial from Fact 6.15. Let

$$Z(y) = p_{d/m}(y_1, \ldots, y_{d/m}) + \cdots + p_{d/m}(y_{d-d/m+1}, \ldots, y_d).$$

Setting $k = m$ in Fact 6.15, let q_m be the resulting polynomial of degree $O(\sqrt{m}\log(1/\gamma))$ and weight $O(m^{\sqrt{m}\log(1/\gamma)})$. Our final polynomial is

$$1 - 2q_m(m - Z(y)).$$

Note that $m - Z(y)$ takes values in $\{0, \ldots, m\}$ and is 0 exactly when all inputs $y_1, \ldots, y_d$ are FALSE. It follows that our final polynomial indeed approximates OR_d to additive error γ on all Boolean inputs.

We bound the degree and weight of this polynomial in y. By Fact 6.14, the inner disjunctions are computed exactly using degree d/m and weight at most 3. Hence, the total degree is $O(\sqrt{m}\log(1/\gamma) \cdot d/m)$. To bound the weight, we observe that the outer polynomial $q_m(\cdot)$ has at most $T = m^{O(\sqrt{m}\log(1/\gamma))}$ terms where each one has degree at most $D_{\text{outer}} = O(\sqrt{m}\log(1/\gamma))$ coefficients of absolute value at most $c_{\text{outer}} = 2^{O(\sqrt{m}\log(1/\gamma))}$. Expanding the polynomials for $Z(y)$, the weight of each term incurs a multiplicative factor of $c_{\text{inner}} \leq 3^{D_{\text{outer}}} = 3^{O(\sqrt{m}\log 1/\gamma)}$ so the total weight is at most $c_{\text{inner}} \cdot c_{\text{outer}} \cdot T = m^{O(\sqrt{m}\log 1/\gamma)}$.

□

Proof of Lemma 6.13. Again we partition the inputs $y_1, \ldots, y_d$ into blocks of size d/m and view the disjunction as:

$$\mathrm{OR}_m(\mathrm{OR}_{d/m}(y_1, \ldots, y_{d/m}), \ldots, \mathrm{OR}_{d/m}(y_{d-d/m+1}, \ldots, y_d)).$$

Once again, we compute the inner disjunctions exactly using the polynomial from Fact 6.14. Let

$$Z(y) = p_{d/m}(y_1, \ldots, y_{d/m}) + \cdots + p_{d/m}(y_{d-d/m+1}, \ldots, y_d).$$

If the input y has Hamming weight at most k, then $Z(y)$ also takes values in $\{m, \ldots, m - 2k\}$. Thus, we may approximate the outer disjunction using a polynomial of degree $O(\sqrt{k}\log(1/\gamma))$ from Fact 6.15. Our final polynomial is:

$$1 - 2q_k(m - Z).$$

The bound on degree and weight may be obtained as in the previous lemma. □

6.4.1 Proof of Theorem 6.1

We now confirm that the results presented so far suffice to establish Theorem 6.1 in the introduction.

Proof of Theorem 6.1. First we take $m = O\big((\log d/\log\log d)^2\big)$ in Lemma 6.12 and take $m = d^{O(1/\sqrt{k})}$ in Lemma 6.13. Combining with Theorem 6.10, it follows that for some constant $C > 0$, the family of d-variate disjunctions restricted to $H_{d,k}$ is 0.01-approximated by the family of d-variate real polynomials of degree t and weight W where

$$t = \min\left\{d^{1-\frac{1}{C\sqrt{k}}}, \frac{d}{\log^{0.995} d}\right\} \text{ and } W = d^{0.01}.$$

Consequently, by Theorem 6.6, we have an algorithm that is a B-iterative database construction where

$$B = O(d^{0.02} t \log d) = O\left(d^{0.02}\log d \cdot \min\left\{d^{1-\frac{1}{C\sqrt{k}}}, \frac{d}{\log^{0.99} d}\right\}\right)$$

and the algorithm runs in time $T = \mathrm{poly}(\binom{d}{\leq t})$

By Theorem 2.15, we have an (ε, δ)-differentially private sanitizer that is $(0.01, 0.01)$-accurate for any set Q of (possibly adaptively chosen) k-way marginal queries provided the size of the database

$$n \geq C \cdot \left(\left(\frac{1}{\varepsilon}\log(100|\mathcal{Q}|)\log\left(\frac{1}{\delta}\right)\right) d^{0.01}\sqrt{\log d} \cdot \min\left\{d^{0.5-\frac{1}{2C\sqrt{k}}}, \frac{d^{0.5}}{\log^{0.48} d}\right\}\right)$$

for an appropriate constant $C > 0$. Further, the algorithm runs in time

$$\mathrm{poly}(n, T) = \mathrm{poly}\left(n, \binom{d}{\leq t}\right) = \mathrm{poly}\left(n, \min\left\{\exp\left(d^{1-1/C\sqrt{k}}\right), \exp\left(d/\log^{.99} d\right)\right\}\right).$$

□

Remark 1 in the introduction follows from using a slightly different choice of m in Lemma 6.12, namely $m = O(\log^2 d/\log^3\log d)$.

To obtain the summary of the database promised in Remark 2, we request an answer to each of the k-way marginal queries $B(1/400)$ times. Doing so, will ensure that we obtain a maximal database update sequence, and it was argued in Section 2.4.3 that the polynomial resulting from any maximal database update sequence accurately answers every k-way marginal query. Finally, we obtain a compact summary by randomly choosing $\widetilde{O}(kd^{.01})$ samples from the normalized coefficient vector of this polynomial to obtain a new sparse polynomial. By a straightforward Chernoff

bound argument, this polynomial will accurately answer every k-way marginal query (see e.g. [19], for an example of this argument in the setting of polynomial approximations). Our compact summary is this final sparse polynomial.

6.5 Limitations of Low-Weight Approximations

In the previous section we saw that the degree required to approximate the function OR_d on inputs of Hamming weight at most k can be significantly smaller than the degree required to do so on all inputs. In this section, we address the question of whether or not the degree can be improved further for inputs of bounded Hamming weight. More generally, we address the general problem of approximating a block-composed function $G = F(\ldots, f(.), \ldots)$, where $F\colon \{-1,1\}^k \to \{-1,1\}$, $f\colon Y \to \{-1,1\}$, $Y \subseteq \mathbb{R}^{d/k}$ over inputs restricted to a set $\mathcal{Y} \subseteq Y^k$ using low-weight polynomials. We give a lower bound on the minimum degree of such polynomials. In our main application, G will equal OR_d, and $\mathcal{Y}$ will be the set of all length d Boolean vectors of Hamming weight at most k.

Our proof technique is inspired by the composition theorem lower bounds shown in [82, Theorem 3.1], where it is shown that the γ-approximate degree of the composed function G is at least the product of the γ-approximate degree of the outer function and the PTF degree of the inner function. Our main contribution is a generalization of such a composition theorem along two directions: (1) we show degree lower bounds that take into account the size of the coefficients of the approximating polynomial, and (2) our lower bounds hold even when we only require the approximation to be accurate on inputs of low Hamming weight, while prior work only considered approximations that are accurate on the entire Boolean hypercube.

Our main theorem is stated below. In parsing the statement of the theorem, it may be helpful to think of $G = \text{OR}_d$, $\mathcal{Y} = H_{d,k}$, the set of all length d Boolean vectors of Hamming weight at most k, $f = \text{OR}_{d/k}$, $F = \text{OR}_k$, $Y = \{-1,1\}^{d/k}$, and $H = H_{d/k,1}$, the set of all Boolean vectors of Hamming weight at most 1. This will be the setting of interest in our main application of the theorem.

Theorem 6.16. *Let $Y \subset \mathbb{R}^{d/k}$ be a finite set and $\gamma > 0$. Given $f : Y \to \{-1,1\}$ and $F : \{-1,1\}^k \to \{-1,1\}$ such that $deg_{2\gamma}(F) = D$, let $G : Y^k \to \{-1,1\}$ denote the composed function defined by $G(Y_1, \ldots, Y_k) = F(f(Y_1), \ldots, f(Y_k))$. Let $\mathcal{Y} \subseteq Y^k$. Suppose there exists*

$H \subseteq Y$ such that for every $(Y_1, \ldots, Y_k) \in Y^k \setminus \mathcal{Y}$ there exists $i \in [k]$ such that $Y_i \in Y \setminus H$. Then, for every $t \in \mathbb{Z}_+$,

$$\textit{deg}_{(\gamma,W)}(G|_{\mathcal{Y}}) \geq \frac{1}{2}tD \textit{ for every } W \leq \gamma 2^{-k} w^*(f|_H, t)^{\frac{D}{2}}.$$

We derive the following corollary from Theorem 6.16. Theorem 6.3 follows immediately from Corollary 6.17 by considering any $k = o(\log d)$.

Corollary 6.17. *Let $k \in [d]$. Then, there exists a universal constant $C > 0$ such that*

$$\textit{deg}_{(1/6,W)}(\textit{OR}_d|_{H_{d,k}}) = \Omega\left(\frac{d}{\sqrt{k}} \cdot \frac{W^{-\frac{1}{C\sqrt{k}}}}{2^{\sqrt{k}/C}}\right).$$

Intuition underlying our proof technique. Recall that our upper bound in Section 6.4 worked as follows. We viewed OR_d as an "OR of ORs", and we approximated the outer OR with a polynomial p of degree $\deg_{\text{outer}}$ chosen to be as small as possible, and composed p with a low-weight but high-degree polynomial computing each inner OR. We needed to make sure the weight W_{inner} of the inner polynomials was very low, because the composition step potentially blows the weight up to roughly $W_{\text{inner}}^{\deg_{\text{outer}}}$. As a result, the inner polynomials had to have very high degree, to keep their weight low.

Intuitively, we construct a dual solution to a certain linear program that captures the intuition that any low-weight, low-degree polynomial approximation to OR_d must look something like our primal solution, composing a low-degree approximation to an "outer" OR with low-weight approximations to inner ORs. Moreover, our dual solution formalizes the intuition that the composition step must result in a massive blowup in weight, from W_{inner} to roughly $W_{\text{inner}}^{\deg_{\text{outer}}}$.

In more detail, our dual construction works by writing OR_d as an OR of ORs, where the outer OR is over k variables, and each inner ORs is over d/k variables. We obtain our dual solution by carefully combining a dual witness Γ to the high approximate degree of the outer OR, with a dual witness ψ to the fact that any low-degree polynomial with margin at least 1 for each inner OR, must have "large" weight, even if the polynomial must satisfy the margin constraint only on inputs of Hamming weight 0 or 1. This latter condition, that ψ must witness high non-constant margin weight even if restricted to inputs of Hamming weight 0 or 1, is essential to ensuring that our combined dual witness does not place any "mass" on irrelevant inputs, i.e. those of Hamming weight larger than k.

6.5.1 Duality Theorems

In the rest of the section, we let $\chi_S(x) = \prod_{i \in S} x_i$ for any given set $S \subseteq [d]$. The question of existence of a weight W polynomial with small degree that γ-approximates a given function can be expressed as a feasibility problem for a linear program. Now, in order to show the non-existence of such a polynomial, it is sufficient to show infeasibility of the linear program. By duality, this is equivalent to demonstrating existence of a solution to the corresponding dual program. We begin by summarizing the duality theorems that will be useful in exhibiting this witness.

Theorem 6.18 (Duality Theorem for (γ, W)-approximate degree)**.** *Fix $\gamma \geq 0$ and let $f : Y \to \{-1, 1\}$ be given for some finite set $Y \subset \mathbb{R}^d$. Then, $\deg_{(\gamma,W)}(f) \geq t + 1$ if and only if there exists a function $\Psi : Y \to \mathbb{R}$ such that*

1. $\sum_{y \in Y} |\Psi(y)| = 1$,
2. $\sum_{y \in Y} \Psi(y) f(y) - W \cdot |\sum_{y \in Y} \Psi(y) \chi_S(y)| > \gamma$ *for every* $S \subseteq [d], |S| \leq t$.

Proof. By definition, $\deg_{(\gamma,W)}(f) \leq t$ if and only if $\exists (\lambda_S)_{S \subseteq [d], |S| \leq t}$:

$$\sum_{S \subseteq [d], |S| \leq t} |\lambda_S| \leq W, \text{ and}$$

$$\left| f(y) - \sum_{S \subseteq [d], |S| \leq t} \lambda_S \chi_S(y) \right| \leq \gamma \; \forall \; y \in Y.$$

By Farkas' lemma, $\deg_{(\gamma,W)}(f) \leq t$ if and only if $\nexists \; \Psi : Y \to \mathbb{R}$ such that

$$\frac{1}{W} \sum_{y \in Y} (f(y)\Psi(y) - \gamma |\Psi(y)|) > \left| \sum_{y \in Y} \chi_S(y) \Psi(y) \right| \; \forall \; S \subseteq [d], |S| \leq t.$$

□

The dual witness that we construct to prove Theorem 6.16 is obtained by combining a dual witness for the large non-constant margin weight of the inner function with a dual witness for the large approximate degree for the outer function. The duality conditions for these are given below. The proof of the duality condition for the case of γ-approximate degree is well-known, and we omit the proof for brevity (see e.g. [83, 92, 20]).

Theorem 6.19 (Duality Theorem for γ-approximate degree)**.** *Fix $\gamma \geq 0$ and let $f : Y \to \{-1, 1\}$ be given, where $Y \subset \mathbb{R}^d$ is a finite set. Then, $deg_\gamma(f) \geq t + 1$ if and only if there exists a function $\Gamma : Y \to \mathbb{R}$ such that*

1. $\sum_{y \in Y} |\Gamma(y)| = 1$,
2. $\sum_{y \in Y} \Gamma(y)p(y) = 0$ *for every polynomial p of degree at most d, and*
3. $\sum_{y \in Y} \Gamma(y)f(y) > \gamma$.

Theorem 6.20 (Duality Theorem for non-constant margin weight)**.** *Let $Y \subset \mathbb{R}^d$ be a finite set, let $f : Y \to \{1, -1\}$ be a given function and $w > 0$. The non-constant margin weight $w^*(f, t) \geq w$ if and only if there exists a distribution $\mu : Y \to [0, 1]$ such that*

1. $\sum_{y \in Y} \mu(y)f(y) = 0$
2. $\left|\sum_{y \in Y} \mu(y)f(y)\chi_S(y)\right| \leq \frac{1}{w}$ *for every $S \subseteq [d]$, $|S| \leq t$.*

Proof. Let $\mathcal{S} = \{S \subseteq [d] : |S| \leq t\}$, $\overline{\mathcal{S}} = \mathcal{S} \setminus \{\emptyset\}$. By definition, $w^*(f, t)$ is expressed by the following linear program:

$$\begin{aligned} \min \sum_{S \in \overline{\mathcal{S}}} |\lambda_S| \\ f(y) \sum_{S \in \mathcal{S}} \lambda_S \chi_S(y) \geq 1 \; \forall \, y \in Y. \end{aligned}$$

The above linear program can be restated as follows:

$$\begin{aligned} \min \sum_{S \in \overline{\mathcal{S}}} \alpha_S \\ \alpha_S + \lambda_S \geq 0 \; \forall \, S \in \overline{\mathcal{S}}, \\ \alpha_S - \lambda_S \geq 0 \; \forall \, S \in \overline{\mathcal{S}}, \\ f(y) \sum_{S \in \mathcal{S}} \lambda_S \chi_S(y) \geq 1 \; \forall \, y \in Y, \text{ and} \\ \alpha_S \geq 0 \; \forall \, S \in \overline{\mathcal{S}}. \end{aligned}$$

The dual program is expressed below:

$$\begin{aligned}
\max \sum_{y} \mu(y) & \\
u_1(S) + u_2(S) &\leq 1 \; \forall \; S \in \overline{\mathcal{S}}, \\
\sum_{y \in Y} \mu(y) f(y) \chi_S(y) + u_1(S) - u_2(S) &= 0 \; \forall \; S \in \overline{\mathcal{S}}, \\
\sum_{y \in Y} \mu(y) f(y) &= 0, \\
\mu(y) \geq 0 \; \forall \; y \in Y, \; u_1(S), u_2(S) &\geq 0 \; \forall \; S \in \overline{\mathcal{S}}.
\end{aligned}$$

By standard manipulations, the above dual program is equivalent to

$$\begin{aligned}
\max \sum_{y} \mu(y) & \\
|\sum_{y \in Y} \mu(y) \chi_S(y) f(y)| &\leq 1 \; \forall \; S \in \overline{\mathcal{S}} \\
\sum_{y \in Y} \mu(y) f(y) &= 0, \\
\mu(y) &\geq 0 \; \forall \; y \in Y
\end{aligned}$$

Finally, given a distribution μ' satisfying the hypothesis of the theorem, one can obtain a dual solution μ to show that $w^*(f,t) \geq w$ with the choice $w^{-1} = \max_{S \in \mathcal{S}} |\sum_{y \in Y} \mu'(y) \chi_S(y) f(y)|$, $\mu(y) = w\mu'(y) \; \forall \; y \in Y$. In the other direction, if $w^*(f,t) \geq w$, then we have a dual solution μ satisfying the above dual program such that $\sum_{y \in Y} \mu(y) = w^*(f,t)$. By setting $\mu'(y) = \mu(y)/w^*(f,t) \; \forall \; y \in Y$, we obtain the desired distribution. □

6.5.2 Proof of Theorem 6.16

Our approach to exhibiting a dual witness as per Theorem 6.18 is to build a dual witness by appropriately combining the dual witnesses for the "hardness" of the inner and outer functions. Our method of combining the dual witnesses is inspired by the technique of [82, Theorem 3.7].

Proof of Theorem 6.16. Let $w = w^*(f|_H, t)$. We will exhibit a dual witness function $\Psi : \mathcal{Y} \to \mathbb{R}$ corresponding to Theorem 6.18 for the specified choice of degree and weight. For $y \in Y^k$, let $Y_i =$

$(y_{(i-1)(d/k)+1}, \ldots, y_{id/k})$. By Theorem 6.20, we know that there exists a distribution $\mu : H \to \mathbb{R}$ such that

$$\sum_{y \in H} \mu(y) f(y) = 0, \tag{6.1}$$

$$\left| \sum_{y \in H} \mu(y) f(y) \chi_S(y) \right| \leq \frac{1}{w} \; \forall \; S \subseteq \left[\frac{d}{k} \right], |S| \leq t \tag{6.2}$$

We set $\mu(y) = 0$ for $y \in Y \setminus H$.

Since $\deg_{2^\gamma}(F) = D$, by Theorem 6.19, we know that there exists a function $\Gamma : \{-1, 1\}^k \to \mathbb{R}$ such that

$$\sum_{x \in \{-1,1\}^k} |\Gamma(x)| = 1, \tag{6.3}$$

$$\sum_{x \in \{-1,1\}^k} \Gamma(x) p(x) = 0 \text{ for every polynomial } p \text{ of degree at most } D, \text{ and} \tag{6.4}$$

$$\sum_{x \in \{-1,1\}^k} \Gamma(x) F(x) > 2\gamma. \tag{6.5}$$

Consider the function $\Psi : Y^k \to \mathbb{R}$ defined as $\Psi(y) = 2^k \Gamma(f(Y_1), \ldots, f(Y_k)) \prod_{i=1}^k \mu(Y_i)$. By the hypothesis of the theorem, we know that if $(Y_1, \ldots, Y_k) \in Y^k \setminus \mathcal{Y}$, then there exists $i \in [k]$ such that $Y_i \in Y \setminus H$ and hence $\mu(Y_i) = 0$ and therefore $\Psi(Y_1, \ldots, Y_k) = 0$.

1.

$$\begin{aligned} \sum_{y \in \mathcal{Y}} |\Psi(y)| &= \sum_{y \in \mathcal{Y}} 2^k |\Gamma(f(Y_1), \ldots, f(Y_k))| \prod_{i=1}^k \mu(Y_i) \\ &= 2^k \mathbb{E}_{y \sim \Phi}(|\Gamma(f(Y_1), \ldots, f(Y_k))|) \end{aligned}$$

where $y \sim \Phi$ denotes y chosen from the product distribution $\Phi : Y^k \to [0, 1]$ defined by $\Phi(y) = \prod_{i \in [k]} \mu(Y_i)$. Since $\sum_{y \in Y} \mu(y) f(y) = 0$, it follows that if Y_i is chosen with probability $\mu(Y_i)$, then $f(Y_i)$ is uniformly distributed in $\{-1, 1\}$. Consequently,

$$\sum_{y \in \mathcal{Y}} |\Psi(y)| = 2^k \mathbb{E}_{z \sim_U \{-1,1\}^k}(|\Gamma(z_1, \ldots, z_k)|) = 1.$$

The last equality is by using (6.3).

2. By the same reasoning as above, it follows from (6.5) that

$$\sum_{y\in\mathcal{Y}} \Psi(y)G(y) = \sum_{z\in\{-1,1\}^k} \Gamma(z)F(z) > 2\gamma.$$

3. Fix a subset $S \subseteq [d]$ of size at most $tD/2$. Let $S_i = S \cap \{(i-1)(d/k)+1, \ldots, id/k\}$ for each $i \in [k]$. Consequently, $\chi_S(y) = \prod_{i=1}^k \chi_{S_i}(Y_i)$.

   Now using the Fourier coefficients $\widehat{\Gamma}(T)$ of the function Γ, we can express

$$\Gamma(z_1, \ldots, z_k) = \sum_{T\subseteq[k]} \widehat{\Gamma}(T) \prod_{i\in T} z_i = \sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \prod_{i\in T} z_i$$

   since $\widehat{\Gamma}(T) = 0$ if $|T| < D$ by (6.4). Hence,

$$\Psi(y) = 2^k \sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \prod_{i\in T} f(Y_i)\mu(Y_i) \cdot \prod_{i\in[k]\setminus T} \mu(Y_i)$$

   Therefore, $\sum_{y\in\mathcal{Y}} \Psi(y)\chi_S(y)$

$$\begin{aligned}
&= \sum_{y\in\mathcal{Y}} \Psi(y) \prod_{i\in[k]} \chi_{S_i}(Y_i) \\
&= 2^k \sum_{y\in\mathcal{Y}} \left(\sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \prod_{i\in T} f(Y_i)\mu(Y_i) \cdot \prod_{i\in[k]\setminus T} \mu(Y_i) \right) \prod_{i\in[k]} \chi_{S_i}(Y_i) \\
&= 2^k \sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \sum_{y\in\mathcal{Y}} \left(\prod_{i\in T} f(Y_i)\mu(Y_i) \cdot \prod_{i\in[k]\setminus T} \mu(Y_i) \prod_{i\in[k]} \chi_{S_i}(Y_i) \right) \\
&= 2^k \sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \sum_{Y_1,\ldots,Y_k\in H} \left(\prod_{i\in T} f(Y_i)\mu(Y_i)\chi_{S_i}(Y_i) \cdot \prod_{i\in[k]\setminus T} \mu(Y_i)\chi_{S_i}(Y_i) \right).
\end{aligned}$$

   Rearranging, we have $\sum_{y\in\mathcal{Y}} \Psi(y)\chi_S(y) =$

$$2^k \sum_{\substack{T\subseteq[k],\\ |T|\geq D}} \widehat{\Gamma}(T) \prod_{i\in T} \left(\sum_{Y_i\in H} f(Y_i)\mu(Y_i)\chi_{S_i}(Y_i) \right) \prod_{i\in[k]\setminus T} \left(\sum_{Y_i\in H} \mu(Y_i)\chi_{S_i}(Y_i) \right). \tag{6.6}$$

Now, we will bound each product term in the outer sum by $w^{-D/2}$. We first observe that for every $i \in [k]$,

$$\sum_{x \in H} \mu(x)\chi_{S_i}(x) \leq \sum_{x \in H} \mu(x) = 1.$$

If $|S_i| \leq t$, by (6.2)

$$\left| \sum_{x \in H} f(x)\mu(x)\chi_{S_i}(x) \right| \leq \frac{1}{w}.$$

If $|S_i| > t$, then

$$\left| \sum_{x \in H} f(x)\mu(x)\chi_{S_i}(x) \right| \leq \sum_{x \in H} \mu(x) = 1.$$

Since $\sum_{i=1}^{k} |S_i| \leq tD/2$, it follows that $|S_i| \leq t$ for more than $k - D/2$ indices $i \in [k]$. Thus, for each $T \subseteq [k]$ such that $|T| \geq D$, there are at least $D/2$ indices $i \in T$ such that $|S_i| \leq t$. Hence,

$$\left| \sum_{y \in \mathcal{Y}} \Psi(y)\chi_S(y) \right| \leq 2^k w^{-\frac{D}{2}} \sum_{\substack{T \subseteq [k], \\ |T| \geq D}} \left| \widehat{\Gamma}(T) \right| \leq 2^k w^{-\frac{D}{2}}.$$

Here, the last inequality is because $|\widehat{\Gamma}(T)| \leq 2^{-k}$ from (6.3).

From 1, 2 and 3, we have

$$\sum_{y \in \mathcal{Y}} \Psi(y)G(y) - W \max_{S \subseteq [d], |S| \leq \frac{tD}{2}} \left| \sum_{y \in \mathcal{Y}} \Psi(y)\chi_S(y) \right| > \gamma$$

if $W \leq \gamma 2^{-k} w^{D/2}$. □

We now derive Corollary 6.17. We need the following theorems on the approximate degree and the non-constant margin weight of the OR_d function.

Theorem 6.21 (OR_d). *[72]* $deg_{1/3}(OR_d) = \Theta(\sqrt{d})$.

Lemma 6.22 (OR_d). $w^*(OR_d|_{H_{d,1}}, t) \geq d/t$.

Proof. The function

$$\mu(x) = \begin{cases} 1/2 & \text{if } x = (1, \ldots, 1), \\ 1/2d & \text{if } x \in H_{d,1} \setminus \{(1, \ldots, 1)\}. \end{cases}$$

acts as the dual witness in Theorem 6.20. □

Proof of Corollary 6.17. We use Theorem 6.16 in the following setting. Let $Y = \{-1,1\}^{d/k}$, the inner function $f : Y \to \{-1,1\}$ be $\mathrm{OR}_{d/k}$ and the outer function $F : \{-1,1\}^k \to \{-1,1\}$ be OR_k, $\mathcal{Y} = H_{d,k}$ and $H = H_{d/k,1}$. By a simple counting argument, if $(Y_1, \ldots, Y_k) \in \{-1,1\}^d \setminus H_{d,k}$, then there exists $i \in [k]$ such that $Y_i \in \{-1,1\}^{d/k} \setminus H_{d/k,1}$. Further, by Theorem 6.21, we know that $\deg_{1/3}(F) = \Theta(\sqrt{k})$ and by Claim 6.22, we know that $w^*(f|_H, t) \geq d/kt$. Therefore, by Theorem 6.16, we have that, for every $t \in \mathbb{Z}_+$,

$$\deg_{1/6,W}(OR_d|_{H_{d,k}}) = \Omega\left(t\sqrt{k}\right) \text{ for every } W \leq \frac{1}{6}2^{-k}\left(\frac{d}{kt}\right)^{C\sqrt{k}}.$$

We obtain the conclusion by taking $t = \lfloor (d/k)(6W2^k)^{-1/C\sqrt{k}} \rfloor$. □

Chapter 7

New Analyst-Private Algorithms

7.1 Introduction

Consider a tracking network that wants to sell a database of consumer data to several competing analysts conducting market research. The administrator of the tracking network faces many opposing constraints when deciding how to provide analysts with this data. The focus of research on differential privacy has been to resolve the dilemma between providing privacy for the data subjects (the consumers in this example) and utility for the data analysts. But the administrator faces another concern—how to provide *privacy for the data analysts*, namely hiding their queries from other data analysts. Indeed, the most interesting databases are going to be analyzed by several competing firms, who will be unwilling to risk disclosing their proprietary strategies.

To address this concern, Dwork, Naor, and Vadhan [33] introduced the notion of *analyst differential privacy.* First, they demonstrated that this concern is valid by showing that any *stateless* sanitizer that answers each query without consideration of the other queries can answer at most $\widetilde{O}(n^2)$ queries with non-trivial accuracy.[15] This result essentially rules out sanitizers that answer a large number of queries while ensuring perfect analyst privacy.[16] Fortunately, they were able to construct a sanitizer that provides a differential-privacy-like guarantee of privacy for the analysts.

[15]The Laplace mechanism is a stateless sanitizer—it adds independent noise to each answer—and answers $\widetilde{\Omega}(n^2)$ queries with non-trivial accuracy (Lemma 2.9).

[16]One can construct a sanitizer that is not stateless but does ensure perfect analyst privacy. However we are not aware of any construction that answers more than $\widetilde{O}(n^2)$ queries, and conjecture that (non-stateless) sanitizers satisfying perfect analyst privacy can answer at most $\widetilde{O}(n^2)$ queries.

In particular, Dwork et al. [33] design a sanitizer with the guarantee that the marginal distribution on answers given to each analyst does not depend significantly on the set of queries asked by all the other analysts. This sanitizer answers a set of counting queries Q on $\{0,1\}^d$ with error $\mathrm{poly}(d, \log|Q|)/n^{1/4}$, and thus can answer exponentially many queries (i.e. $|Q| = 2^{n^{\Omega(1)}}$) with non-trivial accuracy.

However, as they note, their sanitizer has several shortcomings. First, it does not promise differential privacy for the *joint distribution* on answers given to multiple analysts. Therefore, if multiple analysts collude, or if a single malicious analyst interacts with the sanitizer under several identities, then the sanitizer no longer guarantees analyst privacy. Second, their sanitizer achieves a weaker level of accuracy than sanitizers without guarantees of analyst privacy, such as the private multiplicative weights mechanism (Lemma 2.16). That is, analyst privacy is achieved at a cost to accuracy, and it is unknown if this cost is inherent. Finally, their sanitizer can only answer linear queries, rather than arbitrary low-sensitivity queries. See Lemma 2.9 and the surrounding material for the definition of low-sensitivity and results about the Laplace mechanism for arbitrary low-sensitivity queries. Although we have focused on counting queries in this thesis, algorithms similar to the private multiplicative weights algorithm are capable of answering an exponential number of arbitrary low-sensitivity queries with non-trivial accuracy.

In this chapter, we design new analyst-private sanitizers that fully or partially address all of these issues. First, we consider sanitizers that guarantee *one-query-to-many-analyst* privacy—for each analyst a, the joint distribution over answers given to *all other* analysts $a' \neq a$ is differentially private with respect to the change of a single query asked by analyst a. This privacy guarantee is incomparable to the one considered by Dwork, et al. [33]; it is weaker, in that we protect the privacy of only a single query asked by a single analyst (rather than protecting the privacy of all queries asked by multiple analysts), but stronger, in that it protects the privacy of one query against all other analysts, even if they collude (rather than protecting against only a single malicious analyst). Our first result is a sanitizer achieving this notion of analyst privacy, that answers counting queries with error $\mathrm{poly}(d, \log|Q|)/\sqrt{n}$. This dependence on n is optimal up to $\mathrm{polylog}(n)$ factors, even when comparing to sanitizers that only guarantee data privacy.

Our next result is a variant of the first sanitizer that achieves *one-analyst-to-many-analyst* privacy—the sanitizer preserves the privacy of *every* query made by a single analyst, even if *all other* analysts collude. This sanitizer answers counting queries with error $\mathrm{poly}(d, \log|Q|)/n^{1/3}$.

Unfortunately in this setting we don't know how to achieve error approaching $1/\sqrt{n}$, or whether such error is possible under this privacy definition.

These sanitizers operate in the *offline* setting, which is the setting we established in Chapter 2 and have used throughout this thesis. Although we have not discussed the *online* setting in this thesis, algorithms such as private multiplicative weights are capable of answering an exponential number of queries, even if those queries arrive one at a time, and each query must be accurately answered before seeing the remaining queries. Our final result is a sanitizer that satisfies one-query-to-many-analyst privacy and answers counting queries with error $\mathrm{poly}(d, \log|Q|)/n^{2/5}$. This sanitizer also admits a natural extension to answering arbitrary low-sensitivity queries, and can answer arbitrary $(1/n)$-sensitive queries with error $\mathrm{poly}(d, \log|Q|)/n^{1/10}$. We remark that our online algorithms are capable of answering a long *fixed* sequence of queries, but these queries need not be specified in advance. Most online sanitizers without analyst privacy are capable of answering a long *adaptively chosen* sequence of queries, but we do not know how to achieve this property under the constraint of analyst privacy.

When answering k counting queries on a database $D \in (\{0,1\}^d)^n$ all of our algorithms run in time $\widetilde{O}(n \cdot (2^d + k))$. As we've shown in Theorem 3.1, assuming the existence of one-way functions, this running time cannot be improved to $\mathrm{poly}(n, d, k)$ since these algorithms are sanitizers that accurately answer $n^{2+o(1)}$ arbitrary counting queries.

7.1.1 Our Techniques

To prove our results, we take a novel view of private query release as a two player zero-sum game between a *data player* and a *query player*. For each element of the data universe $x \in \{0,1\}^d$, the data player has an action a_x. Recall that a database can be viewed as a probability distribution over the data universe, and that a distribution over the data universe can be viewed as a database consisting of "fractional rows," or alternatively a database of infinite size, and this interpretation is equivalent for the purpose of counting queries. Thus a mixed strategy for the data player can be interpreted as such a database. The goal of the data player is to choose a database that approximates the input database D well.

Given a set of queries Q, we give the query player two actions for each $q \in Q$, a_q and $a_{\neg q}$. The query player tries to play queries for which the data player's approximation to the database performs poorly. The two actions for each query allow the query player to penalize the data player's

approximation both when the approximate answer to q is too high, and when it is too low. Formally, we define the cost matrix by

$$G_{D,Q}(a_q, a_x) = q(x) - q(D) \qquad G_{D,Q}(a_{\neg q}, a_x) = q(D) - q(x).$$

The query player wishes to maximize the cost, whereas the database player wishes to minimize the cost. We show that the value of this game is 0, and that for any pair of *ρ-approximate equilibrium strategies*, the strategy of the data player corresponds to a synthetic database that answers every query $q \in Q$ correctly up to additive error $O(\rho)$.

Different privacy constraints for the private query release problem can be mapped into privacy constraints for solving two-player zero-sum games by defining different notions of adjacency between two cost matrices G, G'. If we consider the standard problem of privately answering a set of counting queries Q, then we want to define G and G' to be adjacent if they differ by at most $1/n$ in every entry, as this is the type of change to G that can arise from changing one row of the input database.

If we consider one-query-to-many-analyst privacy, then we want to define G and G' to be adjacent if they differ arbitrarily on at most two rows of the game matrix, as this is the type of change that can arise if we change one of the input queries in Q. Our main result can be viewed as an algorithm for privately computing an approximate equilibrium of a zero-sum game while protecting the privacy of a two rows in the cost matrix.

To construct an approximate equilibrium, we use a well-known result: when each of the two players in a zero-sum game choose their actions using *no-regret* algorithms, the empirical distribution of play quickly converges to a pair of approximate equilibrium strategies. When we say that a player uses a no-regret algorithm, we mean she has an algorithm that observes the actions of the opposing player and chooses her actions to ensure she cannot achieve a significantly higher payoff in hindsight by playing any single action (or, equivalently, by playing a random choice from a single distribution on actions) at every turn. Thus, to compute an equilibrium of G, we have the query player and the data player play against each other using no-regret algorithms, and output the empirical play distribution of the data player as the synthetic database. We can now discuss the challenges in converting this approach into a differentially private algorithm.

The first problem is that the distribution on actions computed by the no-regret algorithm will not be privacy-preserving on its own, as it necessarily depends on the cost matrix. In previous approach to privately answering counting queries such as [78, 44, 42, 50], which are also based

on no-regret algorithms, privacy has been achieved by perturbing the *inputs* to the no-regret algorithm. Unfortunately, when the inputs contain information about the queries that we now consider sensitive information, there does not appear to be a natural way to perturb the inputs that maintains convergence.

To circumvent this problem, we use the additional structure of a particular no-regret algorithm, the multiplicative weights algorithm. We crucially rely on the fact that *sampling* actions from the distributions maintained by the multiplicative weights algorithm is privacy preserving. Intuitively, privacy will come from the fact that the multiplicative weights algorithm does not adjust the weight on any action too aggressively, meaning that when we view the weights as defining a distribution over actions, changing the losses experienced by the algorithm in various ways will have a limited effect on the distribution over actions. We note that this property is not used in the private multiplicative weights mechanism of Hardt and Rothblum [44], who use the distribution itself as a hypothesis. Indeed, without the constraint of query privacy, *any* no-regret algorithm can be used in place of multiplicative weights [78, 42], which is not the case in our setting. Dwork, Rothblum, and Vadhan [35] used the multiplicative weights algorithm in a similar way in their "private boosting algorithm," however they needed to use a modified form of multiplicative weights for their results. Our techniques can be seen a generalization of their approach.

The second problem is that sampling from the multiplicative weights algorithm only guarantees privacy with respect to a small number of its inputs, which in this case will be the queries chosen by the query player. If the query player chooses the same action repeatedly, then altering one of the input queries could change a large number of inputs to the multiplicative weights algorithm. Intuitively, we must ensure that changing one query from one analyst does not affect the costs experienced by the data player too dramatically. To this end, we force the query player to play mixed strategies from the set of *smooth* distributions, which do not place too much weight on any single action. It is known that playing any no-regret algorithm, but projecting into the set of smooth distributions in the appropriate way (via a *Bregman projection*), will ensure no-regret with respect to any smooth distribution on actions. For comparison, no-regret is typically defined with respect to the best single action, which is a not a smooth distribution on actions. This technique of using smooth distributions was inspired by the work of Barak, Hardt, and Kale [10], and was also used in the results of Dwork et al. [35] on private boosting.

Once we constrain the query player to the set of smooth strategies, we obtain something slightly

weaker than a pair of approximate equilibrium strategies. However, we do obtain a mixed strategy for the data player that ensures a cost no worse than the value of the game as long as the query player does not choose an action from some set of at-most s actions, where $1/s$ is the maximum probability the query player was allowed to place on any single strategy. Moving back to our original goal of answering counting queries, the data player's mixed strategy corresponds to a synthetic database that answers *all but* s queries accurately, and we release this synthetic database to all analysts. Now, the last step in our algorithm is a *cleanup phase*, in which we identify and answer each of the remaining s queries. Since we will chose s to be small, in fact $s = O(n)$, we will be able to identify all of the remaining queries using the sparse vector technique, and then answer each one with Laplacian noise of magnitude $\widetilde{O}(\sqrt{s}/n) = \widetilde{O}(1/\sqrt{n})$. The result is a nearly optimal error rate of $\mathrm{poly}(d, \log|Q|)/\sqrt{n}$.

This approach natural extends to the setting of one-analyst-to-many-analyst privacy by giving the query player one action for each analyst. Here, we will design the cost matrix so that the query player is rewarded for finding an analyst such that at least one query issued by that analyst is not answered well. As before, we will compute a synthetic database that answers the queries of most analysts accurately, and we will have a cleanup phase in which we accurately answer the queries for the small number of remaining analysts.

Finally, in Section 7.6 we use these techniques to convert the private multiplicative weights algorithm of Hardt and Rothblum [44] into an online algorithm that preserves one-query-to-many-analyst privacy, and also answers arbitrary low-sensitivity queries. This algorithm does not directly use the zero-sum game perspective, but does crucially make use of the technique of sampling from the mixed strategy maintained by the multiplicative weights algorithm.

7.2 Preliminaries

The notation we use in this chapter differs slightly from what we have used in Chapters 3-6. These changes are made primarily to improve the clarity of the results by maintaining a clear notational hierarchy. To this end, we will often use $\mathcal{X} = \{0,1\}^d$ to denote the data universe. Additionally, in this algorithm we only consider sanitizers (as opposed to one-shot sanitizers) that take *arbitrary* counting queries as input, thus we will use the calligraphic $\mathcal{Q}$ to refer to the set of input queries and (infrequently) use bold $\mathbf{Q}$ to refer to the set of allowable queries. For comparison, in

the previous chapters we typically used capital Q to denote the set of input queries and calligraphic $\mathcal{Q}$ to refer to the set of allowable queries.

7.2.1 Definitions of Analyst Differential Privacy

To define analyst privacy, we first define *many-analyst sanitizers*. Let $\mathbf{Q}$ be the set of all counting queries. We will assume there are m analysts and will use $I = [m]$ to denote the set of analysts. The input to the sanitizer is a database $D \in (\{0,1\}^d)^n$ and m sets of k queries, $\mathcal{Q}_1, \ldots, \mathcal{Q}_m \in \mathbf{Q}^k$. The assumption that each set contains exactly k queries is for convenience, and has no bearing on the results. The sanitizer returns a sequence $A_1, \ldots, A_m \in \mathbb{R}^k$, where each set A_{id} is understood to be a sequence of answers to the queries asked by some analyst $\mathrm{id} \in I$. Thus, a many-analyst sanitizer has the form $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathbf{Q}^k)^m \to (\mathbb{R}^k)^m$. For each $\mathrm{id} \in I$ we write $\mathcal{M}(D, \mathcal{Q})_{-\mathrm{id}}$ to denote $(A_1, \ldots, A_{\mathrm{id}-1}, A_{\mathrm{id}+1}, \ldots, A_m)$, the output given to all analysts other than id.

Let $\mathcal{Q} = \mathcal{Q}_1, \ldots, \mathcal{Q}_m$ and $\mathcal{Q} = \mathcal{Q}'_1, \ldots, \mathcal{Q}'_m$. We say that $\mathcal{Q}$ and $\mathcal{Q}'$ are *analyst-adjacent* if there exists $\mathrm{id}^* \in I$ such that for every $\mathrm{id} \neq \mathrm{id}^*$, $\mathcal{Q}_{\mathrm{id}} = \mathcal{Q}'_{\mathrm{id}}$. That is, $\mathcal{Q}$ and $\mathcal{Q}'$ are analyst adjacent if they differ only on the queries asked by one analyst. Intuitively, a sanitizer satisfies one-analyst-to-many-analyst privacy if changing *all the queries asked by analyst* id^* does not significantly affect the output given to *all analysts other than* id^*.

Definition 7.1. A many-analyst sanitizer $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathbf{Q}^k)^m \to (\mathbb{R}^k)^m$ satisfies (ε, δ)-*one-analyst-to-many-analyst privacy* if for every database $D \in (\{0,1\}^d)^n$, every two analyst-adjacent query sequences $\mathcal{Q}, \mathcal{Q}'$ that differ only on one set of queries $\mathcal{Q}_{\mathrm{id}}, \mathcal{Q}'_{\mathrm{id}}$, and every $S \subseteq (\mathbb{R}^k)^{m-1}$,

$$\Pr\left[\mathcal{M}(D, \mathcal{Q})_{-\mathrm{id}} \in S\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}(D, \mathcal{Q}')_{-\mathrm{id}} \in S\right] + \delta.$$

Let $\mathcal{Q} = \mathcal{Q}_1, \ldots, \mathcal{Q}_m$ and $\mathcal{Q}' = \mathcal{Q}'_1, \ldots, \mathcal{Q}'_m$. We say that $\mathcal{Q}$ and $\mathcal{Q}'$ are *query-adjacent* if there exists $\mathrm{id}^* \in I$ such that for every $\mathrm{id} \neq \mathrm{id}^*$, $\mathcal{Q}_{\mathrm{id}} = \mathcal{Q}'_{\mathrm{id}}$ and $|\mathcal{Q}_{\mathrm{id}^*} \triangle \mathcal{Q}'_{\mathrm{id}^*}| \leq 1$. That is, $\mathcal{Q}, \mathcal{Q}'$ are query adjacent if they differ only on a single query asked by some analyst id^*. Intuitively, we say that a sanitizer satisfies one-query-to-many-analyst privacy if changing *one query asked by analyst* id^* does not significantly affect the output given to *all analysts other than* id^*.

Definition 7.2. A many-analyst sanitizer $\mathcal{M}\colon (\{0,1\}^d)^n \times (\mathbf{Q}^k)^m \to (\mathbb{R}^k)^m$ satisfies (ε, δ)-*one-query-to-many-analyst privacy* if for every database $D \in (\{0,1\}^d)^n$, every two query-adjacent

query sequences $\mathcal{Q}, \mathcal{Q}'$ that differ only on one query asked by analyst id^*, and every $S \subseteq (\mathbb{R}^k)^{m-1}$,

$$\Pr\left[\mathcal{M}(D, \mathcal{Q})_{-\mathrm{id}^*} \in S\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}(D, \mathcal{Q}')_{-\mathrm{id}^*} \in S\right] + \delta.$$

7.3 Solving Two-Player Zero-Sum Games

7.3.1 (Non-Private) Multiplicative Weights

Let $A \colon \mathcal{A} \to [0, 1]$ be a measure over a set of actions $\mathcal{A}$. We use $|A| = \sum_{a \in \mathcal{A}} A(a)$ to denote the *density* of A. A measure naturally corresponds to a probability distribution $\widetilde{A}$ in which

$$\Pr\left[\widetilde{A} = a\right] = A(a)/|A|$$

for every $a \in \mathcal{A}$. Throughout, we will use calligraphic letters ($\mathcal{A}$) to denote a set of actions, lower case letters (a) to denote the actions, capital letters (A) to denote a measure over actions, and capital letters with a tilde to denote the corresponding distributions ($\widetilde{A}$). We will use the KL-divergence between two distributions, defined to be

$$KL(\widetilde{A}||\widetilde{A}') = \sum_{a \in \mathcal{A}} \widetilde{A}(a) \log\left(\widetilde{A}(a)/\widetilde{A}'(a)\right).$$

Let $L \colon \mathcal{A} \to [0, 1]$ be a loss function (losses L). Abusing notation, we can define $L(A) = \mathbb{E}\left[L(\widetilde{A})\right]$. Given an initial measure A_1, we can define the multiplicative weights algorithm in Figure 16.

For $t = 1, 2, \ldots, T$:
  Sample $a_t \leftarrow_{\mathrm{R}} \widetilde{A}_t$
  Receive losses L_t (may depend on $A_1, a_1, \ldots, A_{t-1}, a_{t-1}$)
  Update:
    For: each $a \in \mathcal{A}$**:**
      Let $A_{t+1}(a) = e^{-\eta L_t(a)} A_t(a)$ for every $a \in \mathcal{A}$

Figure 16: The Multiplicative Weights Algorithm, MW_η

The following theorem about the multiplicative weights update is well-known.

Theorem 7.3 (Multiplicative Weights. See e.g. [74])**.** *Let A_1 be the uniform measure of density 1, and let $\{a_1, \ldots, a_T\}$ be the actions obtained by MW_η with losses $\{L_1, \ldots, L_t\}$. Let $A^* = \mathbf{1}_{a=a^*}$, for some $a^* \in \mathcal{A}$, and $\delta \in (0, 1]$. Then with probability at least $1 - \beta$,*

$$\begin{aligned}\mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(a_t)] &\leq (1+\eta) \mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(A^*)] + \frac{KL(\widetilde{A}^* || \widetilde{A}_1)}{\eta T} + \frac{4\log(1/\beta)}{\sqrt{T}} \\ &\leq \mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(A^*)] + \eta + \frac{\log|\mathcal{A}|}{\eta T} + \frac{4\log(1/\beta)}{\sqrt{T}}.\end{aligned}$$

We need to work with a variant of multiplicative weights that only produces measures A of high density, which will imply that $\widetilde{A}$ does not assign too much probability to any single element of $\mathcal{A}$. To this end, we will apply (a special case of) the Bregman projection to the measures obtained from the multiplicative weights update rule.

Definition 7.4. Let $s \in (0, \mathcal{U}]$. Given a measure A such that $|A| \leq s$, let $\Gamma_s A$ be the *(Bregman) projection of A into the set of density-s measures*, obtained by computing $c \geq 1$ such that $s = \sum_{a \in \mathcal{A}} \min\{1, cA(a)\}$ and setting $\Gamma A(a) = \min\{1, cM(a)\}$ for every $a \in \mathcal{A}$. We call s is the *density* of measure A.

For $t = 1, 2, \ldots, T$:
  Let $A'_t = \Gamma_s A_t$, and sample $a_t \leftarrow_R \widetilde{A}'_t$
  Receive losses L_t (may depend on $A_1, a_1, \ldots, A_{t-1}, a_{t-1}$)
  Update:
    For each $a \in \mathcal{A}$**:**
      Let $A_{t+1}(a) = e^{-\eta L_t(a)} A_t(a)$

Figure 17: The Dense Multiplicative Weights Algorithm, $DMW_{s,\eta}$

Given an initial measure A_1 such that $|A_1| \leq s$, we can define the dense multiplicative weights algorithm in Figure 17. Note that we update the unprojected measure A_t, but sample a_t using the projected measure $\Gamma_s A_t$. Observe that the update step can only decrease the density, so we will have $|A_t| \leq s$ for every t. Like before, given a sequence of losses $\{L_1, \ldots, L_T\}$ and an initial measure A_1 of density s, we can consider the sequence $\{A_1, \ldots, A_T\}$ where A_{t+1} is given by the projected multiplicative weights update applied to A_t, L_t. The following theorem is known.

Theorem 7.5. *Let A_1 be the uniform measure of density 1 and let $\{a_1, \ldots, a_T\}$ be the sequence of measures obtained by $DMW_{s,\eta}$ with losses $\{L_1, \ldots, L_T\}$. Let $A^* = \mathbf{1}_{a \in S*}$ for some set $S^* \subseteq \mathcal{A}$ of size s, and $\delta \in (0, 1]$. Then with probability $1 - \beta$,*

$$\begin{aligned}\mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(\Gamma A_t)] &\leq (1+\eta) \mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(A^*)] + \frac{KL(\widetilde{A}^* || \widetilde{A}_1)}{\eta T} + \frac{4 \log(1/\beta)}{\sqrt{T}} \\ &\leq \mathop{\mathbb{E}}_{t \leftarrow_R [T]} [L_t(A^*)] + \eta + \frac{\log |\mathcal{A}|}{\eta T} + \frac{4 \log(1/\beta)}{\sqrt{T}}.\end{aligned}$$

7.3.2 Regret Minimization and Two-Player Zero-Sum Games

Let $G \colon \mathcal{A}_R \times \mathcal{A}_C \to [0, 1]$ be the cost-matrix for a two-player zero-sum game between two players, (R)ow and (C)olumn, who take actions $r \in \mathcal{A}_R$ and $c \in \mathcal{A}_C$ and receive losses $G(r, c)$ and $-G(r, c)$, respectively. Let $\Delta(\mathcal{A}_R), \Delta(\mathcal{A}_C)$ be the set of measures over actions in $\mathcal{A}_R$ and $\mathcal{A}_C$, respectively. The well-known minimax theorem states that

$$v := \min_{R \in \Delta(\mathcal{A}_R)} \max_{C \in \Delta(\mathcal{A}_C)} G(R, C) = \max_{C \in \Delta(\mathcal{A}_C)} \min_{R \in \Delta(\mathcal{A}_R)} G(R, C).$$

We define this quantity v to be the *value of the game*.

Freund and Schapire [39] showed that if two sequences of actions $\{r_1, \ldots, r_T\}, \{c_1, \ldots, c_T\}$ are "no-regret with respect to one another", then $\widetilde{r} = \frac{1}{T} \sum_{t=1}^{T} r_t$ and $\widetilde{c} = \frac{1}{T} \sum_{t=1}^{T} c_t$ form an approximate equilibrium strategy pair. More formally, if

$$\max_{c \in \mathcal{A}_C} \mathop{\mathbb{E}}_t [G(r_t, c)] - \rho \leq \mathop{\mathbb{E}}_t [G(r_t, c_t)] \leq \min_{r \in \mathcal{A}_R} \mathop{\mathbb{E}}_t [G(r, c_t)] + \rho,$$

then

$$v - 2\rho \leq G(\widetilde{r}, \widetilde{c}) \leq v + 2\rho.$$

Thus, if Row chooses actions using the multiplicative weights update rule with losses $L_t(r_t) = G(r_t, c_t)$ and Column chooses actions using the multiplicative weights rule with losses $L_t(r_t) = -G(r_t, c_t)$, then each player's distribution on actions converges to a minimax strategy. That is, if we play until both players have regret at most ρ:

$$\max_c G(\widetilde{r}, c) \leq v + 2\rho \qquad v - 2\rho \leq \min_r G(r, \widetilde{c}).$$

For query privacy in our view of query release as a two player game, Column must not put too much weight on any single query. Thus, we need an analogue of this result in the case where

Column is not choosing actions according to the multiplicative weights update, but rather using the projected multiplicative weights update. In this case we cannot hope to obtain an approximate minimax strategy, since Column cannot play any single action with significant probability. However, we can define an alternative notion of the value of a game where Column is restricted in this way: let $\Delta_s(\mathcal{A}_C)$ be the set of measures over $\mathcal{A}_C$ of minimum density at least s, and define

$$v_s := \min_{R \in \Delta(\mathcal{A}_R)} \max_{C \in \Delta_s(\mathcal{A}_C)} G(R, C).$$

Notice that $v_s \leq v$, and v_s can be very different from v.

Theorem 7.6. *Let $\{r_1, \ldots, r_T\} \in \mathcal{A}_R$ be a sequence of row-player actions, $\{C_1, \ldots, C_T\} \in \Delta_s(\mathcal{A}_C)$ be a sequence of high-density measures over column-player actions, and $\{c_1, \ldots, c_T\} \in \mathcal{A}_C$ be a sequence of column-player actions such that $c_j \leftarrow_R C_j$ for every $t \in [T]$. Further, suppose that*

$$\mathop{\mathbb{E}}_t [G(r_t, c_t)] \leq \min_{R \in \Delta(\mathcal{A}_R)} \mathop{\mathbb{E}}_t [G(R, c_t)] + \rho \quad \textit{and} \quad \mathop{\mathbb{E}}_t [G(r_t, c_t)] \geq \max_{C \in \Delta_s(\mathcal{A}_C)} \mathop{\mathbb{E}}_t [G(r_t, C)] - \rho.$$

Then,

$$v_s - 2\rho \leq G(\widetilde{r}, \widetilde{c}) \leq v + 2\rho.$$

Moreover, $\widetilde{r}$ is an approximate min-max strategy with respect to strategies in $\Delta_s(\mathcal{A}_C)$, i.e.,

$$v_s - 2\rho \leq \max_{C \in \Delta_s(\mathcal{A}_C)} G(\widetilde{r}, C) \leq v + 2\rho.$$

Proof. For the first set of inequalities, we handle each part separately. For one direction,

$$\begin{aligned} v_s &= \min_{R \in \Delta(\mathcal{A}_R)} \max_{C \in \Delta_s(\mathcal{A}_C)} G(R, C) \\ &\leq \max_{C \in \Delta_s(\mathcal{A}_C)} \mathop{\mathbb{E}}_t [G(r_t, C)] \leq \mathop{\mathbb{E}}_t [G(r_t, c_t)] + \rho \\ &\leq \min_{R \in \Delta(\mathcal{A}_R)} \mathop{\mathbb{E}}_t [G(R, c_t)] + 2\rho = \min_{R \in \Delta(\mathcal{A}_R)} G(R, \widetilde{c}) + 2\rho \\ &\leq G(\widetilde{r}, \widetilde{c}) + 2\rho. \end{aligned}$$

The other direction is similar, starting with the fact that $v = \max_{c \in C} \min_{r \in R} G(r, c)$.

For the second set of inequalities, we also handle the two cases separately. For the upper bound,

$$\begin{aligned} \max_{C \in \Delta_s(\mathcal{A}_C)} \mathop{\mathbb{E}}_t [G(\widetilde{r}, C)] &\leq \mathop{\mathbb{E}}_t [G(r_t, c_t)] + \rho \\ &\leq \min_{R \in \Delta(\mathcal{A}_R)} \mathop{\mathbb{E}}_t [G(R, c_t)] + 2\rho = \min_{R \in \Delta(\mathcal{A}_R)} G(R, \widetilde{c}) + 2\rho \\ &\leq v + 2\rho. \end{aligned}$$

For the lower bound,

$$\max_{C \in \Delta_s(\mathcal{A}_C)} G(\widetilde{r}, C) \geq \mathop{\mathbb{E}}_{t} [G(\widetilde{r}, \widetilde{c})] \geq v_s - 2\rho$$

This completes the proof of the theorem. □

Corollary 7.7. *Let $G\colon \mathcal{A}_R \times \mathcal{A}_C \to [0,1]$. If the row player chooses a sequence of actions $\{r_1, \ldots, r_T\}$ by running MW_η with loss functions $L_t(r) = G(r, c_t)$ and the column player chooses a sequence of actions $\{c_1, \ldots, c_T\}$ by running $DMW_{s,\eta}$ with the loss functions $L_t(c) = -G(r_t, c)$, then with probability at least $1 - \beta$,*

$$v_s - 2\rho \leq \max_{c \in C_s} G(\widetilde{r}, c) \leq v + 2\rho,$$

for

$$\rho = \eta + \frac{\max\{\log|\mathcal{A}_R|, \log|\mathcal{A}_C|\}}{\eta T} + \frac{4\log(2/\beta)}{\sqrt{T}}.$$

7.4 A One-Query-to-Many-Analyst Private Sanitizer

We specify our new sanitizer achieving one-query-to-many-analyst privacy in Figure 18. Recall that we use $\mathcal{X} = \{0,1\}^d$ to specify the algorithm.

7.4.1 Accuracy

Theorem 7.8. *The algorithm in Figure 18 is (α, β)-accurate for*

$$\alpha = O\left(\frac{\sqrt{\log(2^d + 2km)\log(1/\delta)}\log(2km/\beta)}{\varepsilon\sqrt{n}}\right).$$

Proof. Observe that the algorithm computes an approximate equilibrium of the game $G_D(x, q) = \frac{1+q(D)-q(x)}{2}$. Let v, v_s be the value and constrained value of this game, respectively. First, we pin down the quantities v and v_s.

Claim 7.9. *For every D, the value and constrained value of G_D is $1/2$.*

Proof of Claim 7.9. It's clear that the value (and hence constrained value) is at most $1/2$, because

$$\min_x \max_q \frac{1 + q(D) - q(x)}{2} \leq \max_q \frac{1 + q(D) - q(D)}{2} = \frac{1}{2}.$$

Input: Database $D \in \mathcal{X}^n$ and m sets of k counting queries $\mathcal{Q}_1, \ldots, \mathcal{Q}_m$.

Initialize: Let $\mathcal{Q} = \bigcup_{\mathrm{id} \in [m]} \mathcal{Q}_{\mathrm{id}} \cup \neg \mathcal{Q}_{\mathrm{id}}$, let $D_0(x) = 1/|\mathcal{X}|$ for every $x \in \mathcal{X}$, and let $Q_0(q) = 1/|\mathcal{Q}|$ for every $q \in \mathcal{Q}$,

$$T = n \cdot \max\{\log|\mathcal{X}|, \log|\mathcal{Q}|\}, \qquad \eta = \frac{\varepsilon}{4\sqrt{T\log(1/\delta)}}, \qquad s = 12T$$

For: $t = 1, \ldots, T$

DataPlayer:

On input a query $\widehat{q}_t$, for each $x \in \mathcal{X}$:

Update $D_t(x) = D_{t-1}(x) \cdot \exp\left(-\eta\left(\frac{1+\widehat{q}_t(D)-\widehat{q}_t(x)}{2}\right)\right)$

Choose $\widehat{x}_t \leftarrow_{\mathrm{R}} \widetilde{D}_t$ and send $\widehat{x}_t$ to **QueryPlayer**

(Recall that, D_t is a measure and $\widetilde{D}_t$ the distribution corresponding to that measure.)

QueryPlayer:

On input a data element $\widehat{x}_t$, for each $q \in \mathcal{Q}$:

Update $Q_{t+1}(q) = Q_t(q) \cdot \exp\left(-\eta\left(\frac{1+q(D)-q(\widehat{x}_t)}{2}\right)\right)$

Let $P_{t+1} = \Gamma_s Q_{t+1}$

Choose $\widehat{q}_{t+1} \leftarrow_{\mathrm{R}} \widetilde{P}_{t+1}$ and send $\widehat{q}_{t+1}$ to **DataPlayer**

(Recall that, P_{t+1} is a measure and $\widetilde{P}_{t+1}$ the corresponding distribution.)

End For:

GenerateSynopsis:

Let $\widehat{D} = (\widehat{x}_1, \ldots, \widehat{x}_T)$ be a synthetic database.

For a parameter $\alpha_{\widehat{D}} > 0$ to be chosen later, run the sparse vector algorithm on the set of queries $F = \left\{ f_q(D) = |q(D) - q(\widehat{D}) \mid q \in \mathcal{Q} \right\}$ to obtain $3s$ queries $\mathcal{Q}_{\text{final}} \subseteq \mathcal{Q}$.

Run Laplace mechanism $\mathcal{M}_{\mathrm{Lap}(\sigma)}(D, \mathcal{Q}_{\text{final}})$ to obtain an answer a_q for each $q \in \mathcal{Q}_{\text{final}}$

Output $\widehat{D}$ to all analysts and, for each $q \in \mathcal{Q}_{\text{final}}$, output (q, a_q) to the appropriate analyst.

Figure 18: A One-Query-to-Many-Analyst Private Sanitizer

Suppose we choose x such that $(1+q(D)-q(x))/2 < 1/2$ for some $q \in \mathcal{Q}$. Then, since the query $q' = 1-q$ is also in $\mathcal{Q}$, $(1+q'(D)-q'(x))/2 > 1/2$. But then $\max_{q\in\mathcal{Q}}(1+q(D)-q(x))/2 > 1/2$, so the value of the game is at least $1/2$.

For the constrained value, suppose we choose x such that $\mathbb{E}_{q\leftarrow_{\mathrm{R}}\mathcal{Q}}\left[(1+q(D)-q(x))/2\right] < 1/2$ for some s-smooth distribution on queries $\mathcal{Q}$. Then we can flip the output of every query in $\mathcal{Q}$ to get a new distribution $\mathcal{Q}'$ such that $\mathbb{E}_{q\leftarrow_{\mathrm{R}}\mathcal{Q}'}\left[(1+q(D)-q(x))/2\right] > 1/2$. So $v_s \geq 1/2$ as well. $\square$

Let $\widehat{D} = \frac{1}{T}\sum_{t=1}^{T} x_t$. By Corollary 7.7,

$$v_s - 2\rho \leq \max_{Q\in\Delta_s(\mathcal{Q})}\left(\frac{1}{2}\mathop{\mathbb{E}}_{q\leftarrow_{\mathrm{R}}\widetilde{Q}}\left[1+q(D)-q(\widehat{D})\right]\right) \leq v + 2\rho.$$

Applying Claim 7.9 and rearranging terms, with probability at least $1-\beta/3$,

$$\begin{aligned}
&\left|\max_{Q\in\Delta_s(Q)}\left(\mathop{\mathbb{E}}_{q\leftarrow_{\mathrm{R}}\widetilde{Q}}\left[q(D)-q(\widehat{D})\right]\right)\right| = \max_{Q\in\Delta_s(\mathcal{Q})}\left(\mathop{\mathbb{E}}_{q\leftarrow_{\mathrm{R}}\widetilde{Q}}\left[\left|q(D)-q(\widehat{D})\right|\right]\right) \leq 4\rho \\
&= 4\left(\eta + \frac{\max\{\log|\mathcal{X}|, \log|\mathcal{Q}|\}}{\eta T} + \frac{4\log(2/\beta)}{\sqrt{T}}\right) \\
&= O\left(\frac{\sqrt{\log(|\mathcal{X}|+|\mathcal{Q}|)\log(1/\delta)} + \log(1/\beta)}{\varepsilon\sqrt{n}}\right) := \alpha_{\widehat{D}}.
\end{aligned}$$

The previous statement suffices to show that $|q(D)-q(\widetilde{D})| \leq \alpha_{\widehat{D}}$ for all but s queries. Otherwise, the uniform distribution over the bad queries would be a distribution over queries contained in $\Delta_s(\mathcal{Q})$, with expected error larger than $\alpha_{\widehat{D}}$.

We can now run the sparse vector algorithm (Lemma 2.11). With probability at least $1-\beta/3$, it will identify every query q with error larger than $\alpha_{\widehat{D}} + \alpha_{SV}$ for

$$\alpha_{\mathrm{SV}} = O\left(\frac{\sqrt{s\log(1/\delta)}\log(\mathcal{Q}/\beta)}{\varepsilon n}\right).$$

Since there are at most s such queries, with probability at least $1-\beta/3$, the Laplace mechanism (Lemma 2.11) answers these queries to within error

$$\alpha_{\mathrm{Lap}} = O\left(\frac{\sqrt{s\log(1/\delta)}\log(s/\beta)}{\varepsilon n}\right).$$

Now, observe that in the final output, there are two ways that a query can be answered: either by $\widehat{D}$, in which case its answer can have error as large as $\alpha_{\widehat{D}}+\alpha_{\mathrm{SV}}$, or by the Laplace mechanism, in which

case its answer can have error as large as α_{Lap}. Thus, with probability at least $1-\beta$, every query has error at most $\max\{\alpha_{\widehat{D}} + \alpha_{\text{SV}}, \alpha_{\text{Lap}}\}$. Substituting our choice of $s = 12T = O(n \cdot \log(|\mathcal{X}| + |\mathcal{Q}|))$ and simplifying, we conclude that the sanitizer is (α, β)-accurate for

$$\alpha = O\left(\frac{\sqrt{\log(|\mathcal{X}| + |\mathcal{Q}|)\log(1/\delta)}\log(|\mathcal{Q}|/\beta)}{\varepsilon\sqrt{n}}\right).$$

□

7.4.2 Data Privacy

Theorem 7.10. *The algorithm in Figure 18 satisfies (ε, δ)-differential privacy for the database.*

Before proving the theorem, we will state a useful lemma about the Bregman projection onto the set of high density measures (Definition 7.4).

Lemma 7.11 (Projection Preserves Differential Privacy)**.** *Let $A_0, A_1 \colon \mathcal{A} \to [0,1]$ be two full-support measures over a set of actions $\mathcal{A}$ and $s \in (0, |\mathcal{A}|)$ be such that $|A_0|, |A_1| \leq s$ and $|\ln(A_0(a)/A_1(a))| \leq \varepsilon$ for every $a \in \mathcal{A}$. Let $A_0' = \Gamma_s A_0$ and $A_1' = \Gamma_s A_1$. Then we have that $|\ln(A_0'(a)/A_1'(a))| \leq 2\varepsilon$ for every $a \in \mathcal{A}$.*

Proof of Lemma 7.11. Recall that to compute $A' = \Gamma_s A$, we find a "scaling factor" $c > 1$ such that

$$\sum_{a \in \mathcal{A}} \min\{1, cA(a)\} = s,$$

and set $A'(a) = \min\{1, cA(a)\}$. Let c_0 and c_1 be the scaling factors for A_0' and A_1' respectively. Assume without loss of generality that $c_0 \leq c_1$. First, observe that

$$\left|\ln\left(\frac{\min\{1, c_0 A_0(a)\}}{\min\{1, c_0 A_1(a)\}}\right)\right| \leq \left|\ln\left(\frac{A_0(a)}{A_1(a)}\right)\right| \leq \varepsilon,$$

for every $a \in \mathcal{A}$. Second, we observe that $c_1/c_0 \leq e^{\varepsilon}$. If this were not the case, then we would have $c_1 A_1(a) \geq c_0 A_1(a) e^{\varepsilon} \geq c_0 A_0(a)$ for every $a \in \mathcal{A}$, with strict inequality for at least one a. But then,

$$\sum_{a \in \mathcal{A}} \min\{1, c_1 A_1(a)\} > \sum_{a \in \mathcal{A}} \min\{1, c_0 A_0(a)\} = s,$$

which would contradict the choice of c_1. Thus,

$$\left|\ln\left(\frac{\min\{1, c_0 A_0(a)\}}{\min\{1, c_1 A_1(a)\}}\right)\right| \leq \left|\ln\left(\frac{\min\{1, c_0 A_0(a)\}}{\min\{1, c_0 A_1(a)\}}\right)\right| + \left|\ln\left(\frac{c_1}{c_0}\right)\right| \leq \varepsilon + \varepsilon = 2\varepsilon,$$

for every $a \in \mathcal{A}$. □

Now we prove the main result of this section.

Proof of Theorem 7.10. We focus on analyzing the privacy properties of the synthetic database $\widehat{D} = (\widehat{x}_1, \ldots, \widehat{x}_T)$, the privacy of the final stage of the sanitizer will follow from the privacy analysis of the sparse vector algorithm and the Laplace mechanism. We will actually show the stronger guarantee that the sequence $v = (\widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_T, \widehat{q}_T)$ is differentially private for the data. Fix a pair of adjacent databases $D_0 \sim D_1$ and let V_0, V_1 denote the distribution on sequences v when the sanitizer is run on database D_0, D_1 respectively. We will show that with probability at least $1 - \delta/3$ over $v = (\widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_T, \widehat{q}_T) \leftarrow_{\mathrm{R}} V_0$,

$$\left| \ln \left(\frac{V_0(v)}{V_1(v)} \right) \right| \leq \frac{\varepsilon}{3},$$

which is no weaker than $(\varepsilon/3, \delta/3)$-differential privacy. To do so, we analyze the privacy of each element of v, $\widehat{x}_t$ or $\widehat{q}_t$, and apply the composition analysis of Dwork, Rothblum, and Vadhan [35]. Define $\varepsilon_0 = 2\eta T/n$.

Claim 7.12. *For every v, and every $t \in [T]$,*

$$\left| \ln \left(\frac{V_0(\widehat{x}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_{t-1}, \widehat{q}_{t-1})}{V_1(\widehat{x}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_{t-1}, \widehat{q}_{t-1})} \right) \right| \leq \varepsilon_0.$$

Proofof Claim 7.12. We can prove the statement by the following direct calculation.

$$\left| \ln \left(\frac{V_0(\widehat{x}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_{t-1}, \widehat{q}_{t-1})}{V_1(\widehat{x}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_{t-1}, \widehat{q}_{t-1})} \right) \right| = \left| \ln \left(\frac{\exp\left(-(\eta/2)\sum_{j=1}^{t-1} 1 + \widehat{q}_j(D_0) - \widehat{q}_j(\widehat{x}_t)\right)}{\exp\left(-(\eta/2)\sum_{j=1}^{t-1} 1 + \widehat{q}_j(D_1) - \widehat{q}_j(\widehat{x}_t)\right)} \right) \right|$$

$$= \frac{\eta}{2} \left| \left(\sum_{j=1}^{t-1} 1 + \widehat{q}_j(D_0) - \widehat{q}_j(\widehat{x}_t) \right) - \left(\sum_{j=1}^{t-1} 1 + \widehat{q}_j(D_1) - \widehat{q}_j(\widehat{x}_t) \right) \right|$$

$$= \frac{\eta}{2} \left| \sum_{j=1}^{t-1} \widehat{q}_j(D_0) - \widehat{q}_j(D_1) \right| \leq \frac{\eta(t-1)}{2n} \leq \frac{\eta T}{2n} \leq \varepsilon_0$$

□

Claim 7.13. *For every v, and every $t \in [T]$,*

$$\left| \ln \left(\frac{V_0(\widehat{q}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_t)}{V_1(\widehat{q}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_t)} \right) \right| \leq \varepsilon_0.$$

Proof of Claim 7.13. The sample $\widehat{q}_t$ is made according to $\widetilde{P}_t$, which is the distribution corresponding to the projected measure P_t. First we'll look at the unprojected measure Q_t. Observe that, for any database D and query q,

$$Q_t(q) = \exp\left(-(\eta/2)\sum_{j=1}^{t-1} 1 + q(D) - q(\widehat{x}_j)\right).$$

Thus, if $Q_0(q)$ is the measure we would have when database D_0 is the input, and $Q_1(q)$ is the measure we would have when database D_1 is the input, then

$$\left|\ln\left(\frac{Q_0(q)}{Q_1(q)}\right)\right| \le \frac{\eta}{2}\left|\sum_{j=1}^{t-1} q_j(D_0) - q_j(D_1)\right| \le \frac{\eta T}{2n},$$

for every $q \in \mathcal{Q}$. Given that Q_0 and Q_1 satisfy this condition, Lemma 7.11 guarantees that the projected measures satisfy

$$\left|\ln\left(\frac{P_0(q)}{P_1(q)}\right)\right| \le \frac{\eta T}{n}.$$

Finally, we note that if the above condition is satisfied for every $q \in \mathcal{Q}$, then the distributions $\widetilde{P_0}, \widetilde{P_1}$ satisfy

$$\left|\ln\left(\frac{\widetilde{P}_0(q)}{\widetilde{P}_1(q)}\right)\right| \le \frac{2\eta T}{n} \le \varepsilon_0,$$

because the value of the normalizer also changes by at most a multiplicative factor of $e^{\pm\eta T/n}$. We observe that $V_b(\widehat{q}_t \mid \widehat{x}_1, \widehat{q}_1, \ldots, \widehat{x}_t) = \widetilde{P}_b(\widehat{q}_t)$ for $b \in \{0,1\}$, which completes the proof of the claim. □

Now, the composition lemma (Theorem 2.3) (for $2T$-fold composition) guarantees that with probability at least $1 - \delta/3$,

$$\left|\ln\left(\frac{V_0(v)}{V_1(v)}\right)\right| \le \varepsilon_0\sqrt{4T\log(3/\delta)} + 4\varepsilon_0^2 T,$$

which is at most $\varepsilon/3$ by our choice of ε_0. This implies that $\widehat{D}$ is $(\varepsilon/3, \delta/3)$-differentially private.

We can choose the parameters of the sparse vector computation so that the set of queries $\mathcal{Q}_{\text{final}}$ is $(\varepsilon/3, \delta/3)$-differentially private, and we can choose the parameters of the Laplace mechanism so that the answers to the queries in $\mathcal{Q}_{\text{final}}$ are $(\varepsilon/3, \delta/3)$-differentially private. □

7.4.3 Analyst Privacy

Theorem 7.14. *The algorithm in Figure 18 satisfies (ε, δ)-one-query-to-many-analyst differential privacy.*

Before query privacy, we will state a useful composition lemma. The lemma is a generalization of the "secrecy of the sample lemma" [51, 35] to the interactive setting. Consider the following game:

- Fix an (ε, δ)-differentially private algorithm $\mathcal{A}\colon \mathcal{U}^* \to \mathcal{R}$ and a bit $b \in \{0, 1\}$. Let $D_0 = \emptyset$.
- For $t = 1, \ldots, T$:
  - The (randomized) adversary $\mathcal{B}(y_1, \ldots, y_t; r)$ chooses two distributions B_t^0, B_t^1 such that $SD(B_t^0, B_t^1) \leq \sigma$.
  - Choose $x_t \leftarrow_{\text{R}} B_t^b$ and let $D_t = D_{t-1} \cup \{x_t\}$.
  - Choose $y_t \leftarrow_{\text{R}} \mathcal{A}(D_t)$.

For a fixed algorithm $\mathcal{A}$ and adversary $\mathcal{B}$, let V^0 be the distribution on $(y_1, \ldots, y_T)$ when $b = 0$ and V^1 be the distribution on $(y_1, \ldots, y_T)$ when $b = 1$.

Lemma 7.15. *If $\varepsilon \leq 1/2$ and $T\sigma \leq 1/12$, then with probability at least $1 - T\delta - \delta'$ over $y = (y_1, \ldots, y_T) \leftarrow_R V^0$,*

$$\left| \ln \left(\frac{V^0(y)}{V^1(y)} \right) \right| \leq \varepsilon(T\sigma)\sqrt{2T \log(1/\delta')} + 30\varepsilon^2(T\sigma)T.$$

We prove this lemma in Section 7.7.

We also need another lemma about the Bregman projection onto the set of high-density measures (Definition 7.4)

Lemma 7.16. *Let $A_0\colon \mathcal{A} \to [0, 1]$ and $A_1\colon \mathcal{A} \cup \{a^*\} \to [0, 1]$ be two full-support measures over their respective sets of actions and $s \in (0, |\mathcal{A}|)$ be such that 1) $|A_0|, |A_1| \leq s$ and 2) $A_0(a) = A_1(a)$ for every $a \in \mathcal{A}$. Let $A_0' = \Gamma_s A_0$ and $A_1' = \Gamma_s A_1$. Then $SD(\widetilde{A}_0', \widetilde{A}_1') \leq 1/s$.*

Proofof Lemma 7.16. Using the form of the projection (Definition 7.4), it is not hard to see that for $a \neq a^*$, $A_0'(a) \geq A_1'(a)$. For convenience, we will write $A_0'(a^*) = 0$ even though a^* is technically

outside of the domain of A'_0. We can now show the following.

$$\begin{aligned}
\sum_{a \in \mathcal{A} \cup \{a^*\}} |A'_0(a) - A'_1(a)| &= |A'_0(a^*) - A'_1(a^*)| + \sum_{a \neq a^*} |A'_0(a) - A'_1(a)| \\
&\leq 1 + \sum_{a \neq a^*} |A'_0(a) - A'_1(a)| \\
&= 1 + \sum_{a \neq a^*} A'_0(a) - A'_1(a) \qquad (A'_0(a) \geq A'_1(a) \text{ for } a \neq a^*) \\
&= 1 + |A'_0| - (|A'_1| - A'_1(a^*)) \leq 1 + |A'_0| - (|A'_1| - 1) \\
&= 1 + s - (s-1) = 2
\end{aligned}$$

We also have that $|A'_0| = |A'_1| = s$, so

$$\begin{aligned}
SD(\widetilde{A}'_0, \widetilde{A}'_1) &= \frac{1}{2} \sum_{a \in \mathcal{A} \cup \{a^*\}} \left| \frac{A'_0(a)}{|A'_0|} - \frac{A'_1(a)}{|A'_1|} \right| \\
&= \frac{1}{2s} \sum_{a \in \mathcal{A} \cup \{a^*\}} |A'_0(a) - A'_1(a)| \leq \frac{1}{s}.
\end{aligned}$$

□

Now we can prove one-query-to-many-analyst privacy.

Proof of Theorem 7.14. Fix a database D. Consider two query-adjacent query sets $\mathcal{Q}_0, \mathcal{Q}_1$ and, without loss of generality assume $\mathcal{Q}_0$ be $\mathcal{Q}_1$ with an arbitrary query by some analyst id replaced with q^*. We write the output to all analysts as $v = (\widehat{x}_1, \ldots, \widehat{x}_T, b_1, \ldots, b_{|\mathcal{Q}|}, a_1, \ldots, a_{|\mathcal{Q}|})$ where $\widehat{D} = \{\widehat{x}_1, \ldots, \widehat{x}_T\}$ is the database that is released to all analysts, $b_1, \ldots, b_{|\mathcal{Q}|}$ is a sequence of bits that indicates whether or not $q(\widehat{D})$ is close to $q(D)$, and $a_1, \ldots, a_{|\mathcal{Q}|}$ is a sequence of approximate answers to the queries $q(D)$ (or $\perp$, if $q(\widehat{D})$ is already accurate). We write $v_{-\mathrm{id}}$ for the portion of v that excludes outputs specific to analyst id's queries. Let V_0, V_1 be the distribution on outputs when the query set is $\mathcal{Q}_0$ and $\mathcal{Q}_1$, respectively.

We analyze the three parts of v separately. First we show that $\widehat{D}$, which is shared among all analysts, satisfies analyst privacy.

Claim 7.17. *With probability at least $1 - \delta$ over the samples $\widehat{x}_1, \ldots, \widehat{x}_T \leftarrow_R V_0$,*

$$\left| \ln \left(\frac{V_0(\widehat{x}_1, \ldots, \widehat{x}_T)}{V_1(\widehat{x}_1, \ldots, \widehat{x}_T)} \right) \right| \leq \varepsilon/2.$$

Proofof Claim 7.17. To prove the claim, we show how the output $\widehat{x}_1, \ldots, \widehat{x}_T$ can be viewed as the output of an instantiation of the algorithm analyzed by Lemma 7.15. For every $t \in [T]$ and $\widehat{q}_1, \ldots, \widehat{q}_{t-1}$, we define the measure D_t over database items to be

$$D_t(x) = \exp\left(-(\eta/2)\sum_{j=1}^{t-1} 1 + \widehat{q}_j(D) - \widehat{q}_j(x)\right).$$

Notice that if we replace a single query $\widehat{q}_\ell$ with $\widehat{q}'_\ell$ and obtain the measure D'_t, then for every $x \in \mathcal{X}$,

$$\left|\ln\left(\frac{\widetilde{D}_t(x)}{\widetilde{D}'_t(x)}\right)\right| \leq \eta.$$

Thus we can view $\widehat{x}_t$ as the output of an η-differentially private algorithm $\mathcal{A}_D(\widehat{q}_1, \ldots, \widehat{q}_{t-1})$, which fits into the framework of Lemma 7.15. (Here, $\widehat{x}_t$ plays the role of y_t and $\widehat{q}_1, \ldots, \widehat{q}_{t-1}$ plays the role of D_{t-1} in the description of the game, while the input database D is part of the description of $\mathcal{A}$).

Now, in order to apply Lemma 7.15, we need to argue the distribution on samples $\widehat{q}_t$ when the query set is $\mathcal{Q}_0$ is *statistically close* to the distribution on samples $\widehat{q}_t$ when the query set is $\mathcal{Q}_1$. Fix any $t \in [T]$ and let Q_0, Q_1 be the measure Q_t over queries maintained by the query player when the input query set is $\mathcal{Q}_0, \mathcal{Q}_1$, respectively. For $q \neq q^*$, we have

$$Q_0(q) = Q_1(q) = \exp\left(-(\eta/2)\sum_{j=1}^{t-1} 1 + q(D) - q(\widehat{x}_j)\right).$$

Additionally, we set $Q_0(q^*) = 0$ (for notational convenience), while $Q_1(q^*) \in (0, 1]$. Thus, if we let $P_0 = \Gamma_s Q_0$ and $P_1 = \Gamma_s Q_1$, we will have $SD(\widetilde{P}_0, \widetilde{P}_1) \leq 1/s$ by Lemma 7.16. Since the statistical distance is $1/s = 1/12T$, we can apply Lemma 7.15 to show that with probability at least $1 - \delta$,

$$\left|\ln\left(\frac{V_0(\widehat{x}_1, \ldots, \widehat{x}_T)}{V_1(\widehat{x}_1, \ldots, \widehat{x}_T)}\right)\right| \leq \frac{\eta\sqrt{T\log(1/\delta)}}{8} + \frac{5\eta^2 T}{2} \leq \frac{\varepsilon}{2}. \qquad (\eta = \varepsilon/4\sqrt{T\log(1/\delta)})$$

□

Now that we have shown $\widehat{D}$ satisfies $(\varepsilon/2, \delta)$-one-query-to-many-analyst differential privacy, it remains to show that the remainder of the output satisfies perfect one-query-to-many-analyst privacy. Recall from the proof of Theorem 7.8 that $\widehat{D}$ will be accurate for all but s queries. That is, if we let $\{f_q\}_{q\in\mathcal{Q}}$ consist of the functions $f_q(D) = |q(D) - q(\widehat{D})|$, then

$$|\{j \mid f_j(D) \geq \alpha\}| \leq s,$$

where α is chosen as in Theorem 7.8. By Lemma 2.11, the sparse vector algorithm will release bits $b_1, \ldots, b_{|\mathcal{Q}|}$ (the indicator vector of the subset of queries with large error) such that for every $\mathrm{id} \in I$, the distribution on b_{-id} is $(\varepsilon/2, 0)$-differentially private with respect to a change in the queries corresponding to any analyst $\mathrm{id}' \neq \mathrm{id}$. Finally, for each query q_j such that $b_j = 1$, the output to the owner of that query will include $a_j = q_j(D) + z_j$ where z_j is an independent sample from the Laplace distribution. These outputs do not depend on any other query, and thus are perfectly one-query-to-many analyst private. This completes the proof of the theorem.

□

7.5 A One-Analyst-to-Many-Analyst Private Sanitizer

In this section we present an algorithm for answering counting queries that satisfies the stronger notion of one-analyst-to-many-analyst privacy. The algorithm is similar to that of Figure 18, but with two notable modifications.

First, instead of the "query player," we will have an "analyst player" who chooses analysts as actions and is trying to find an analyst $\mathrm{id} \in [m]$ for which there is at least one query in $\mathcal{Q}_{\mathrm{id}}$ with large error (recall that the queries are given to the algorithm in sets $\mathcal{Q}_1, \ldots, \mathcal{Q}_m$). That is, the analyst player attempts to find $\mathrm{id} \in [m]$ to maximize $\max_{q \in \mathcal{Q}_{\mathrm{id}}} q(D) - q(\widehat{D})$.

Second, we will compute a database $\widehat{D}$ such that $\max_{q \in \mathcal{Q}_{\mathrm{id}}} |q(D) - q(\widehat{D})|$ is small for all but s *analysts* in the set $[m]$, rather than having the s mishandled queries. We can still use sparse vector to find these s analysts, however we can't answer the queries with the Laplace mechanism, since each of the analysts may ask an exponential number of queries. However, since there are not too many analysts remaining, we can use s independent copies of the multiplicative weights mechanism (each run with $\varepsilon' \approx \varepsilon/\sqrt{s}$) to handle each analyst's queries.

7.5.1 Accuracy

Theorem 7.18. *The algorithm in Figure 19 is (α, β)-accurate for*

$$\alpha = \widetilde{O}\left(\frac{\sqrt{\log(2^d + m)\log k}\log(m/\beta)\log^{3/4}(1/\delta)}{\varepsilon n^{1/3}}\right).$$

Input: Database $D \in \mathcal{X}^n$, and m sets of k linear queries $\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_m$. For $\mathrm{id} \in I = [m]$, let $\mathcal{Q}_{\mathrm{id}} = \overline{\mathcal{Q}}_{\mathrm{id}} \cup \neg\overline{\mathcal{Q}}_{\mathrm{id}}$.

Initialize: Let $D_0(x) = 1/|\mathcal{X}|$ for each $x \in \mathcal{X}$, let $I_0(q) = 1/m$ for each $\mathrm{id} \in I$,

$$T = n^{2/3} \max\{\log|\mathcal{X}|, m\}, \qquad \eta = \frac{\varepsilon}{4\sqrt{T\log(1/\delta)}}, \qquad s = 12T.$$

DataPlayer:

On input an analyst $\widehat{\mathrm{id}}_t$, for each $x \in \mathcal{X}$, update:

$$D_t(x) = D_{t-1}(x) \cdot \exp\left(-\eta \max_{q \in \mathcal{Q}_{\widehat{\mathrm{id}}_t}} \left(\frac{1 + \widehat{q}_t(D) - \widehat{q}_t(x)}{2}\right)\right)$$

Choose $\widehat{x}_t \leftarrow_{\mathrm{R}} \widetilde{D}_t$ and send $\widehat{x}_t$ to **AnalystPlayer**

AnalystPlayer:

On input a data element $\widehat{x}_t$, for each $\mathrm{id} \in \mathcal{I}$, update:

$$I_{t+1}(\mathrm{id}) = I_t(\mathrm{id}) \cdot \exp\left(-\eta \max_{q \in \mathcal{Q}_{\mathrm{id}}} \left(\frac{1 + q(D) - q(\widehat{x}_t)}{2}\right)\right)$$

Let $P_{t+1} = \Gamma_s I_{t+1}$

Choose $\widehat{\mathrm{id}}_{t+1} \leftarrow_{\mathrm{R}} \widetilde{P}_{t+1}$ and send $\widehat{\mathrm{id}}_{t+1}$ to **DataPlayer**

GenerateSynopsis:

Let $\widehat{D} = (\widehat{x}_1, \ldots, \widehat{x}_T)$ be a synthetic database.

For a parameter $\alpha_{\widehat{D}} > 0$ to be chosen later, run the sparse vector algorithm on the set of queries $F = \left\{ f_{\mathrm{id}}(D) = \max_{q \in \mathcal{Q}_{\mathrm{id}}} |q(D) - q(\widehat{D}) \mid \mathrm{id} \in I \right\}$ to obtain a set of at most $3s$ analysts $I_{\text{final}} \subseteq I$.

For each analyst $\mathrm{id} \in I_{\text{final}}$, run $\mathcal{M}_{\mathrm{MW}}(D, \mathcal{Q}_{\mathrm{id}})$ with parameters $\varepsilon' = \frac{\varepsilon}{10\sqrt{s\log(3s/\delta)}}$ and $\delta' = \frac{\delta}{3s}$ to obtain a set of answers $A^{(\mathrm{id})}$

Output $\widehat{D}$ to all analysts and f each $\mathrm{id} \in I_{\text{final}}$, output A_{id} to analyst id.

Figure 19: A One-Analyst-to-Many-Analyst Private Sanitizer

Proof. As we discussed above, the algorithm is computing an approximate equilibrium of the game

$$G_{D,\mathcal{Q}}(x, \mathrm{id}) = \max_{\mathrm{id} \in [m]} \max_{q \in \mathcal{Q}_{\mathrm{id}}} \frac{1 + q(D) - q(x)}{2}.$$

Let v, v_s be the value and constrained value of this game, respectively. First we pin down the quantities v and v_s.

Claim 7.19. *For every $D, m, \mathcal{Q}$, the value and constrained value of $G_{D,m,\mathcal{Q}}$ is $1/2$.*

The proof of this claim is omitted, but is nearly identical to that of Claim 7.9.

Let $\widehat{D} = \frac{1}{T}\sum_{t=1}^{T} \widehat{x}_t$. By Corollary 7.7,

$$v_s - 2\rho \leq \max_{I \in \Delta_s([m])} \mathop{\mathbb{E}}_{\mathrm{id} \leftarrow_{\mathrm{R}} \widetilde{I}} \left[\max_{q \in \mathcal{Q}_{\mathrm{id}}} \left(\frac{1 + q(D) - q(\widehat{D})}{2} \right) \right] \leq v + 2\rho.$$

Applying Claim 7.19 and rearranging terms, we have that with probability $1 - \beta/3$,

$$\begin{aligned}
&\left| \max_{I \in \Delta_s([m])} \left(\mathop{\mathbb{E}}_{\mathrm{id} \leftarrow_{\mathrm{R}} \widetilde{I}} \left[\max_{q \in \mathcal{Q}_{\mathrm{id}}} q(D) - q(\widehat{D}) \right] \right) \right| = \max_{I \in \Delta_s([m])} \left(\mathop{\mathbb{E}}_{\mathrm{id} \leftarrow_{\mathrm{R}} \widetilde{I}} \left[\max_{q \in \mathcal{Q}_{\mathrm{id}}} \left| q(D) - q(\widehat{D}) \right| \right] \right) \leq 4\rho \\
&= 4\left(\eta + \frac{\max\{\log|\mathcal{X}|, \log m\}}{\eta T} + \frac{4\log(3/\beta)}{\sqrt{T}} \right) \\
&= O\left(\frac{\sqrt{\log(|\mathcal{X}| + m)\log(1/\delta)} + \log(1/\beta)}{\varepsilon n^{1/3}} \right) := \alpha_{\widehat{D}}.
\end{aligned}$$

The previous statement suffices to show that $\max_{q \in \mathcal{Q}_{\mathrm{id}}} |q(D) - q(\widetilde{D})| \leq \alpha_{\widehat{D}}$ for all but s analysts $\mathrm{id} \in I$. Otherwise, the uniform distribution over the analysts for which the error bound of $\alpha_{\widehat{D}}$ does not hold would be a distribution over analysts, contained in $\Delta_s([m])$ with expected error larger than $\alpha_{\widehat{D}}$.

Since there are at most s such analysts we can run the sparse vector algorithm (Lemma 2.11), and, with probability at least $1 - \beta/3$, it will identify every analyst id such that the maximum error over all queries in $\mathcal{Q}_{\mathrm{id}}$ is larger than $\alpha_{\widehat{D}} + \alpha_{SV}$ for

$$\alpha_{\mathrm{SV}} = O\left(\frac{\sqrt{s\log(1/\delta)}\log(m/\beta)}{\varepsilon n} \right).$$

There are at most s such analysts. Thus, running the private multiplicative weights algorithm (Lemma 2.16) independently for each of these analysts' queries—with privacy parameters $\varepsilon' =$

$\Theta(\varepsilon/\sqrt{s\log(s/\delta)})$ and $\delta' = \Theta(\delta/s)$—will yield answers such that, with probability $1-\beta/3$, for every $\mathrm{id} \in I'$,

$$\begin{aligned}\max_{q\in\mathcal{Q}_{\mathrm{id}}} |q(D) - a_q| &\le O\left(\frac{s^{1/4}\log^{1/4}|\mathcal{X}|\sqrt{\log(s|\mathcal{Q}_{\mathrm{id}}|/\beta)}\log^{3/4}(s/\delta)}{\sqrt{\varepsilon n}}\right)\\ &\le \widetilde{O}\left(\frac{n^{1/6}\sqrt{\log(|\mathcal{X}|+m)\log(|\mathcal{Q}_{\mathrm{id}}|/\beta)}\log^{3/4}(1/\delta)}{\sqrt{\varepsilon n}}\right)\\ &\le \widetilde{O}\left(\frac{\sqrt{\log(|\mathcal{X}|+m)\log(|\mathcal{Q}_{\mathrm{id}}|/\beta)}\log^{3/4}(1/\delta)}{n^{1/3}\sqrt{\varepsilon}}\right) := \alpha_{\mathrm{MW}}.\end{aligned}$$

Taking a union bound, observing that the maximum error on any query is $\max\{\alpha_{\widehat{D}} + \alpha_{\mathrm{SV}}, \alpha_{\mathrm{MW}}\}$, and simplifying, we get that the sanitizer is (α,β)-accurate for

$$\alpha = \widetilde{O}\left(\frac{\sqrt{\log(|\mathcal{X}|+m)\log|\mathcal{Q}_{\mathrm{id}}|}\log(m/\beta)\log^{3/4}(1/\delta)}{\varepsilon n^{1/3}}\right).$$

□

7.5.2 Data Privacy

Theorem 7.20. *The algorithm in Figure 19 satisfies (ε,δ)-differential privacy for the data.*

We omit the proof of this theorem, which follows that of Theorem 7.10 almost identically. The only difference is that in the final step, we need to argue that running s independent copies of multiplicative weights with privacy parameters $\varepsilon' = \Theta(\varepsilon/\sqrt{s\log(s/\delta)})$ and $\delta' = \Theta(\delta/s)$ satisfies $(\varepsilon/3,\delta/3)$-differential privacy, which follows directly from the composition properties of differential privacy (Theorem 2.3).

7.5.3 Query Privacy

In this section we prove query privacy for our one analyst to many analyst sanitizer.

Theorem 7.21. *The algorithm in Figure 19 satisfies (ε,δ)-one-analyst-to-many-analyst differential privacy.*

Proof. Fix a database D. Consider two analyst-adjacent sets of queries $\mathcal{Q}_0, \mathcal{Q}_1$. Without loss of generality assume $\mathcal{Q}_0$ is just $\mathcal{Q}_1$ with all the queries by some analyst id replaced with a new

set $\mathcal{Q}'^{(\mathrm{id})}$. We write the output to all analysts as $v = (\widehat{x}_1, \ldots, \widehat{x}_T, b_1, \ldots, b_m, A_1, \ldots, A_m)$ where $\widehat{D} = \widehat{x}_1, \ldots, \widehat{x}_T$ is the database that is released to all analysts, $b_1, \ldots, b_m$ is a sequence of bits that indicates whether or not $q_j(\widehat{D})$ is close to $q_j(D)$ for every $q \in \mathcal{Q}_{\mathrm{id}}$, and $A_1, \ldots, A_m$ is a sequence consisting of the output of the multiplicative weights mechanism for every analyst $\mathrm{id} \in I$ and $\perp$ for every other analyst. Let V_0, V_1 be the distribution on outputs when the queries are $\mathcal{Q}_0$ and $\mathcal{Q}_1$, respectively.

The proof closely follows the proof of one-query-to-many-analyst privacy for Algorithm 3. Showing that the final two parts b, A of the output are query private is essentially the same, so we will focus on proving that $\widehat{D}$ satisfies one-analyst-to-many-analyst privacy.

Claim 7.22. *With probability at least* $1 - \delta$ *over* $\widehat{x}_1, \ldots, \widehat{x}_T \leftarrow_R V_0$,

$$\left| \ln \left(\frac{V_0(\widehat{x}_1, \ldots, \widehat{x}_T)}{V_1(\widehat{x}_1, \ldots, \widehat{x}_T)} \right) \right| \leq \varepsilon.$$

Proofof Claim 7.22. To prove the claim, we show how the output $\widehat{x}_1, \ldots, \widehat{x}_T$ can be viewed as the output of an instantiation of the sanitizer analyzed by Lemma 7.15. Notice that for every $t \in [T]$ and $\widehat{\mathrm{id}}_1, \ldots, \widehat{\mathrm{id}}_{t-1}$, we can write the measure D_t over database items as

$$D_t(x) = \exp\left(-(\eta/2) \sum_{j=1}^{t-1} \max_{q \in Q^{(\widehat{\mathrm{id}}_j)}} 1 + \widehat{q}_j(D) - \widehat{q}_j(x) \right).$$

If we replace a single analyst $\widehat{\mathrm{id}}_\ell$ with $\widehat{\mathrm{id}}'_\ell$, and obtain the measure D'_t, then for every $x \in \mathcal{X}$,

$$\left| \ln \left(\frac{\widetilde{D}_t(x)}{\widetilde{D}'_t(x)} \right) \right| \leq \eta.$$

Thus we can view $\widehat{x}_t$ as the output of an η-differentially private algorithm $\mathcal{M}_D(\widehat{\mathrm{id}}_1, \ldots, \widehat{\mathrm{id}}_{t-1})$, which fits into the framework of Lemma 7.15. (Here, $\widehat{x}_t$ plays the role of y_t and $\widehat{\mathrm{id}}_1, \ldots, \widehat{\mathrm{id}}_{t-1}$ plays the role of D_{t-1} in the description of the game, while the input database D is part of the description of $\mathcal{A}$).

As before, we apply Lemma 7.15, to argue that the distribution on analysts $\widehat{\mathrm{id}}_t$ when the query set is $\mathcal{Q}_0$ is statistically close to the distribution on analysts $\widehat{\mathrm{id}}_t$ when the analyst set is $\mathcal{Q}_1$. The argument does not change significantly, thus we can apply Lemma 7.15 to show that with probability at least $1 - \delta$,

$$\left| \ln \left(\frac{V(\widehat{x}_1, \ldots, \widehat{x}_T)}{V'(\widehat{x}_1, \ldots, \widehat{x}_T)} \right) \right| \leq \frac{\eta \sqrt{T \log(1/\delta)}}{8} + \frac{5\eta^2 T}{2} \leq \frac{\varepsilon}{2}. \qquad (\eta = \varepsilon/(4\sqrt{T \log(1/\delta)}))$$

□

As before, the remainder of the output satisfies perfect one-analyst-to-many-analyst privacy. This completes the proof of the theorem. □

7.6 Another One-Query-to-Many-Analyst Private Sanitizer

So far we have presented two analyst-private sanitizers capable of answering exponentially many queries. These sanitizers were in the offline setting, which has been the focus of the thesis. These sanitizers were also limited in that they were only capable of answering counting queries, whereas variants of the median mechanism [78] can answer exponentially many arbitrary low-sensitivity queries while ensuring differential privacy.

In this section, we present a new sanitizer satisfying one-query-to-many-analyst privacy that addresses these issues. Since we have only introduced notation and terminology for the offline setting, we will present and analyze our sanitizer in that setting. However, it will be clear from the construction that the sanitizer would be just as effective if the queries were specified online, so long as the entire sequence of queries is fixed in advance. That is, the sequence need not be known to the sanitizer ahead of time, but may not be chosen adaptively in response to the answers given by the sanitizer. In contrast, many non-query-private online sanitizers are capable of answering an exponentially long sequence of *adaptively chosen* queries. See [44] for a more formal treatment of the online setting. Although we will only describe this sanitizer formally in the case of counting queries, in Section 7.6.4 we will discuss how it can be extended to the case of arbitrary low-sensitivity queries.

The new sanitizer is similar to the multiplicative weights algorithm of Hardt and Rothblum [44]. In their algorithm, a "hypothesis" about the true database is maintained throughout the sequence of queries. In particular, the hypothesis will be a probability distribution over the data universe—essentially an infinite database. When a query arrives, it is classified according to whether or not the current hypothesis accurately answers that query. If it does, then the query is answered according to the hypothesis. Otherwise, the query is answered with a noisy answer computed from the true database and the hypothesis is updated using the multiplicative weights update rule.

The main challenge in making that algorithm query-private is to argue that the current hypothesis does not depend too much on the previous queries. We overcome this difficulty by "sampling from the hypothesis" to obtain a finite database that represents the hypothesis well, but satisfies

query-privacy. We must balance the need to take many samples—so that the database we obtain by sampling accurately reflects the hypothesis database, and the need to limit the impact of any one query on the sampled database. To handle both these constraints, we introduce *batching*—instead of updating every time we find a query not well-answered by the hypothesis, we batch together s queries at a time, and do one update on the average of these queries to limit the influence of any single query.

A note on terminology: the execution of the algorithm takes place in several rounds, where each round processes one query. Rounds where the query is answered using the real database are called *bad rounds*; rounds that are not bad are *good rounds*. We will split the rounds into T *epochs* and the distributions D_t and hypotheses H_t correspond to the t-th epoch.

7.6.1 Accuracy

First, we sketch a proof that the online sanitizer answers counting queries accurately. Intuitively, there are three ways that our algorithm might give an inaccurate answer, and we treat each separately. First, in a good round, the answer given by the hypothesis may be a bad approximation to the true answer. Second, in a bad round, the answer given may have too much noise. We address these two cases with straightforward arguments showing that the noise is not too large in any round.

The third way the algorithm may be inaccurate is if there are more than R bad rounds, and the algorithm terminates early. We show that this is not the case using a potential argument: after sufficiently many bad rounds, the hypothesis D_T and the sample H_T will be accurate for all queries in the stream, and thus there will be no more bad rounds. The potential argument is a natural extension of the argument in Hardt and Rothblum [44] to handle the additional error coming from taking samples from D_t to obtain H_t.

Theorem 7.23. *The algorithm in Figure 20 is* (α, β)*-accurate for*

$$\alpha = O\left(\frac{\log^{3/10}|\mathcal{X}|\sqrt{\log(|\mathcal{Q}|/\beta)\log(1/\delta)}}{\varepsilon^{3/2}n^{2/5}}\right).$$

Proof. First we condition on the event that the magnitude of the noise is sufficiently small in every round. Specifically,

$$\forall i \in [k],\ |z_i| \leq \sigma \log(4k/\beta). \tag{7.1}$$

Input: Database $D \in \mathcal{X}^n$, sequence $\mathcal{Q} = \{q_1, \ldots, q_k\}$ of counting queries

Initialize: $D_0(x) = 1/|\mathcal{X}|$ for each $x \in \mathcal{X}$, $H_0 = D_0$, $\mathcal{U}_0 = \emptyset$, $s_0 = s + \mathrm{Lap}(2/\varepsilon)$, $t = 0$, $r = 0$,

$$\eta = \frac{2\sqrt{\log(4|\mathcal{Q}|/\beta)\log(1/\delta)}\log^{3/10}|\mathcal{X}|}{\varepsilon^{3/5}n^{2/5}}, \qquad \widehat{n} = \sqrt{\frac{4\log(4|\mathcal{Q}|/\beta)}{\tau^2}}, \qquad T = \frac{\log|\mathcal{X}|}{\eta^2},$$

$$s = \frac{2\sqrt{8\widehat{n}T\log(1/\delta)}}{\varepsilon}, \qquad R = 2sT, \qquad \sigma = \frac{4\sqrt{R}\log(1/\delta)}{\varepsilon n},$$

$$\tau = \frac{16\sqrt{\log(4|\mathcal{Q}|/\beta)\log(1/\delta)}\log^{3/10}|\mathcal{X}|}{\varepsilon^{3/5}n^{2/5}}.$$

AnswerQueries:

While $t < T, r < R, i \le k$, on input query q_i**:**

Let $z_i = \mathrm{Lap}(\sigma)$

If $|q_i(D) - q_i(H_t) - z_i| \le \tau$**: Output:** $q_i(H_t)$

Else:

Let $u = \mathrm{sgn}(q_i(D) - q_i(H_t) + z_i) \cdot q_i$ and let $\mathcal{U}_t = \mathcal{U}_t \cup \{u\}$, $r = r + 1$

Output: $q_i(D) + z_i$

If $|\mathcal{U}_t| > s_t$**:**

Let $(D_{t+1}, H_{t+1}) = \mathbf{Update}(D_t, \mathcal{U}_t)$

Let $\mathcal{U}_{t+1} = \emptyset$, and let $s_{t+1} = s + \mathrm{Lap}(2/\varepsilon)$ and let $r = 0$.

Advance to query q_{t+1}

Update:

Input: Distribution D_t, update queries $\mathcal{U}_t = \{u_1, \ldots, u_{s_t}\}$

For each $x \in \mathcal{X}$**:**

Let $\mathbf{u}_t(x) = \frac{1}{s}\sum_{j=1}^{s_t} u_j(x)$ and update $D_{t+1}(x) = e^{(\eta/2)\mathbf{u}_t(x)} \cdot D_t(x)$.

Normalize D_{t+1} and let H_{t+1} be $\widehat{n}$ independent samples from D_{t+1}

Return: (D_{t+1}, H_{t+1})

Figure 20: Analyst-Private Multiplicative Weights for Counting Queries

A standard analysis of the tails of the Laplace distribution shows that (7.1) holds with probability at least $1 - \beta/4$.

Next we show that, conditioned on (7.1), the algorithm answers every query accurately so long as it has not terminated before answering all k queries.

Claim 7.24. *Assume* (7.1) *holds. Then prior to termination of the algorithm, every query is answered with error at most* $\tau + \sigma \log(4k/\beta)$

Proofof Claim 7.24. First we consider bad rounds. In these rounds q_i is answered with $q_i(D) + z_i$. Conditioned on (7.1), all of these queries are answered sufficiently accurately.

Now we consider good rounds. In these rounds we answer with $q_i(H_t)$, and we will only have a good round if $|q_i(D) - q_i(H_t) - z_i| \leq \tau$. Conditioned on (7.1), we can only have a good round if $|q_i(D) - q_i(H_t)| \leq \tau + \sigma \log(4k/\beta)$. □

Now we must show that the algorithm does not terminate early. Recall that it can terminate early either because it hits a limit on the number of epochs, or because it hits a limit on the number of bad rounds. We will use a potential argument to show that there cannot be too many epochs. The number of bad rounds that is in epoch t is a random variable s_t, and we will also show that with high probability, there are not too many bad rounds within the T epochs.

In doing the analysis, it will be useful to establish the property that the values of s_t are close to s on average. Let $S_t = s_t - s$, be the value of the noise added to s in the t-th epoch. We want to condition on the event

$$|S| \leq \frac{8\sqrt{T}\log(4/\beta)}{\varepsilon} \leq \frac{T}{2}, \tag{7.2}$$

where the final equality holds for large enough T, so long as ε and β are not too small as a function of T. We can easily deduce that event (7.2) holds with probability at least $1 - \beta/2$ from the following lemma.

Lemma 7.25 ([42])**.** *Let* $X_1, \ldots, X_T$ *be* T *independent draws from* $\mathrm{Lap}(2/\varepsilon)$*, and let* $X = \sum_{t=1}^{T} X_t$*. Then,*

$$\Pr\left[|X| > \frac{8\sqrt{T}\log(2/\beta)}{\varepsilon}\right] < \beta.$$

We also want to establish that, in every epoch, H_t is "close" to D_t in the sense that

$$\forall i \in [k],\ |q_i(D_t) - q_i(H_t)| \leq \tau/4, \tag{7.3}$$

where t is the epoch in which the i-th query is asked. Recall that H_t consists of $\widehat{n}$ random samples from the distribution D_t. Thus, a standard Chernoff bound will establish that $|q_i(D_t) - q_i(H_t)| \leq \sqrt{4\log(4k/\beta)/\widehat{n}} \leq \tau/4$ with probability at least $1 - \beta/4$.

We will now demonstrate that, with high probability, the algorithm does not terminate early.

Claim 7.26. *Assume* (7.1)*,* (7.2)*, and* (7.3) *hold. Then the algorithm does not terminate before answering all k queries.*

Proofof Claim 7.26. We will use a potential argument on the sequence of distributions D_t. The potential function will be

$$\Phi_t = KL(D||D_t) := \sum_{x \in \mathcal{X}} D(x) \log\left(\frac{D(x)}{D_t(x)}\right).$$

Elementary properties of the potential function show that $\Phi_t \geq 0$ and $\Phi_0 = KL(D||D_0) \leq \log|\mathcal{X}|$. The decrease in potential from epoch to epoch can be expressed in terms of the error of the current distribution on the update query.

Lemma 7.27 (See e.g. [44])**.**

$$\Phi_{t-1} - \Phi_t \geq \eta\left(\mathbf{u}_t(D) - \mathbf{u}_t(D_{t-1})\right) - \eta^2.$$

Since the potential function is bounded between 0 and $\log|\mathcal{X}|$, we can get a bound on the number of epochs by showing that the potential decreases significantly between most epochs. Given the preceding lemma, we simply need to show that the queries $\mathbf{u}_1, \mathbf{u}_2, \ldots$ have large (positive) error.

Recall that $\mathbf{u}_t = \frac{1}{s}\sum_{u \in \mathcal{U}_t} u$. Also recall that if $u \in \mathcal{U}$ and $u = q_i$, then the reason q_i is in $\mathcal{U}$ is because $q_i(D) - q_i(H_{t-1}) + z_i > \tau$. Similarly, if $u = \neg q_i$, then $q_i(D) - q_i(H_{t-1}) + z_i < -\tau$. We will focus on the first case where $q_i(D) - q_i(H_{t-1}) + z_i > \tau$, the other case will follow similarly. We can get a lower bound on $u(D) - u(D_{t-1})$ as follows.

$$\begin{aligned}
u(D) - u(D_{t-1}) &= (u(D) - u(H_{t-1}) + z_i) + (q_i(H_{t-1}) - q_i(D_{t-1}) + z_i) \\
&\geq (u(D) - u(H_{t-1}) + z_i) - |z_i| - |q_i(H_{t-1}) - q_i(D_{t-1})| \\
&\geq \tau - |z_i| - |q_i(H_{t-1}) - q_i(D_{t-1})|
\end{aligned}$$

We need to show that the right-hand side of the final expression is large. By (7.1), we have that $|z_i| \leq \sigma \log(4k/\beta) \leq \tau/4$. By (7.3), we have that $|q_i(H_{t-1}) - q_i(D_{t-1})| \leq \tau/4$. So we have stablished that for every $u \in \mathcal{U}_t$,

$$u(D) - u(D_{t-1}) \geq \tau - \tau/4 - \tau/4 = \tau/2.$$

We can now express the error on $\mathbf{u}_t$ in terms of $|\mathcal{U}_t|$ as

$$\mathbf{u}_t(D) - \mathbf{u}_t(D_{t-1}) = \frac{1}{s} \sum_{u \in \mathcal{U}_t} u(D) - u(D_{t-1}) \geq \frac{\tau |\mathcal{U}_t|}{2s}.$$

In turn, we get an expression for the total decrease in potential after T epochs in terms of $\sum_{t \leq T} |\mathcal{U}_t|$, and can apply (7.2) to lower bound the total decrease in potential.

$$\begin{aligned}
\Phi_T - \Phi_0 &\geq \frac{\eta\tau}{2s} \left(\sum_{t \leq T} |\mathcal{U}_t| \right) - T\eta^2 \\
&= \frac{\eta\tau}{2s} \left(\sum_{t \leq T} s_t \right) - T\eta^2 = \frac{\eta\tau T}{2} + \frac{\eta\tau}{2s} \left(\sum_{t \leq T} S_t \right) - T\eta^2 = \frac{\eta\tau T}{4} - T\eta^2.
\end{aligned}$$

Now, noting that $\tau \geq 8\eta$, we have

$$\Phi_T - \Phi_0 \geq 2\eta^2 T - \eta^2 T \geq \eta^2 T.$$

Thus, conditioning on all the events above, $T \leq \log |\mathcal{X}|/\eta^2 \leq n^{4/5} \log |\mathcal{X}|$. Thus we have shown that the algorithm will not terminate by reaching its limit of epochs. Finally, we must show that the algorithm does not terminate because it reaches its limit of bad rounds. Assuming (7.2), the number of bad rounds is at most

$$\sum_{t=1}^{T} s_t = \sum_{t=1}^{T} (s + S_t) \leq 2Ts.$$

Thus, assuming (7.1), (7.2), and (7.3), the algorithm does not terminate due to reaching its limit on the number of epochs or bad rounds, and indeed answers all k queries prior to terminating. □

By a union bound, the probability that all of (7.1), (7.2), and (7.3) hold is at least $1 - \beta$. Thus, combining the two claims proves the theorem. □

7.6.2 Data Privacy

In this section we establish that our sanitizer satisfies differential privacy. As discussed in the introduction, we will rely on the generic blueprint of $\mathcal{M}_{\mathrm{IDC}}$—the combination of sparse vector with an IDC (Section 2.4.3).

Theorem 7.28. *The algorithm in Figure 20 satisfies (ε, δ)-differential privacy.*

Proof sketch. We will simply outline how to cast our algorithm as the combination of an IDC with the sparse vector algorithm. There is some ambiguity caused by our use of the term "update" within the specification of the algorithm. Here, the different periods for the IDC correspond to the intervals between *bad* rounds (rather than *update* rounds). The approximation maintained in each period is the state $(D_t, H_t, \mathcal{U}_t, s_t, r)$. The updates have two types, if $|\mathcal{U}_t| < s_t$ then the update is simply to add the query u to $\mathcal{U}_t$. If $|\mathcal{U}_t| = s_t$, then the update is to perform a multiplicative weights computation and resample the hypothesis H_t and reset the other parameters. Notice that the number of update rounds, in this sense, is at most $B = TR$. Thus, we can verify that the parameters for the surrounding sparse vector algorithm are set appropriately to ensure (ε, δ)-differential privacy. □

7.6.3 Query Privacy

More interestingly, we show that this sanitizer satisfies one-query-to-many-analyst privacy.

Theorem 7.29. *The algorithm in Figure 20 is (ϵ, δ)-one-query-to-many-analyst private.*

Proof. Fix the input database D and the values of the Laplace noise, $z_1, \ldots, z_k$. We will show that for every value of the Laplace random variables, the sanitizer satisfies analyst privacy. Consider any two adjacent sequences of queries $\mathcal{Q}_0, \mathcal{Q}_1$. It will be sufficient to restrict to the case where $\mathcal{Q}_0 = q_1, \ldots, q_k$ and $\mathcal{Q}_1 = q^*, q_1, \ldots, q_k$. If the query on which $\mathcal{Q}_0$ and $\mathcal{Q}_1$ differ is not the first query in the sequence, then the portion of the transcript of the sanitizer that answers the common prefix of queries will reveal no information about whether the query sequence is $\mathcal{Q}_0$ or $\mathcal{Q}_1$.

We want to argue that the answers to *all queries in* $\mathcal{Q}$ are private, but *not* that the answer to q^* is private (if it is requested). We will represent the answers to the queries in $\mathcal{Q}$ by a sequence $\{(H_t, i_t)\}_{t \in [T]}$ where H_t is the hypothesis used in the t-th epoch and i_t is the index of the last query in that epoch (the one that caused the sanitizer to switch to hypothesis H_t). Observe that for a fixed database D, Laplace noise, and sequence of queries $\mathcal{Q}$, we can simulate the output of the

sanitizer *for all queries in* $\mathcal{Q}$ given only this information—once we fix a hypothesis H_t, we can determine whether any query q will be added to the update set $\mathcal{U}_t$ for this epoch. So once we begin epoch t with hypothesis H_t, we have fixed all the bad rounds, and once we are given i_t, we have determined when epoch t ends and epoch $t+1$ begins. At this point, we fix the next hypothesis H_{t+1} and continue simulating.

Formally, let V_0, V_1 be distribution over sequences $\{(H_t, i_t)\}$ when the query sequence is $\mathcal{Q}_0, \mathcal{Q}_1$, respectively. We will show that with probability at least $1-\delta$, if $\{(H_t, i_t)\}_{t\in[T]}$ is drawn from V_0, then

$$\left|\ln\left(\frac{V_0(\{(H_t,i_t)\})}{V_1(\{(H_t,i_t)\})}\right)\right| \le \varepsilon.$$

Recall that $\mathcal{U}_t$ is the set of queries that are used to update the distribution D_t to D_{t+1}. We will use $\mathcal{U}_{\le t} = \bigcup_{j=0}^{t} \mathcal{U}_t$ to denote the set of all queries used to update the distributions $D_0, \ldots, D_t$. Notice that if q^* does not get added to the set $\mathcal{U}_0$, then V_0 and V_1 will be distributed identically. Therefore, suppose $q^* \in \mathcal{U}_0$. First we must reason about the joint distribution of the first component of the output.

Claim 7.30. *For all* H_0, i_0*,*

$$\left|\ln\left(\frac{V_0(H_0,i_0)}{V_1(H_0,i_0)}\right)\right| \le \frac{\varepsilon}{2}.$$

Proofof Claim 7.30. Since H_0 does not depend on the query sequence, it will be identically distributed in both cases. Indeed, H_0 is simply $\widehat{n}$ random samples from the uniform distribution over $\mathcal{X}$. Once H_0 is fixed, we can determine whether a query q will cause an update. Fix query q_{i_0} and assume that it is the s-th update query in the sequence $q_1, \ldots, q_k$ and the $(s+1)$-st update query in the sequence $q^*, q_1, \ldots, q_k$. Then $V_0(i_0|H_0) = \Pr[s_0 = s]$ and $V_1(i_0|H_0) = \Pr[s_0 = s+1]$. By the basic properties of the Laplace distribution,

$$\left|\ln\left(\frac{V_0(i_0|H_0)}{V_1(i_0|H_0)}\right)\right| \le \frac{\varepsilon}{2}.$$

□

Now we reason about the remaining components $(H_1, i_1), \ldots, (H_T, i_T)$.

Claim 7.31. *For every H_0, i_0, with probability at least $1-\delta$ over the choice of the sequence of components $v = (H_1, i_1, \ldots, H_T, i_T) \leftarrow_R (V_0 \mid v_{t-1})$, we have*

$$\left|\ln\left(\frac{V_0(v \mid H_0, i_0)}{V_1(v \mid H_0, i_0)}\right)\right| \leq \frac{\varepsilon}{2}.$$

Proofof Claim 7.31. We will show that v is the $\widehat{n}T$-fold composition of $(\varepsilon_0, 0)$-differentially private algorithms for suitable ε_0. Fix a prefix $v_{t-1} = H_0, i_0, \ldots, H_{t-1}, i_{t-1}$. Given this prefix, we can determine for any given sequence of queries $q_1, \ldots, q_{i_{t-1}}$ or $q^*, q_1, \ldots, q_{i_{t-1}}$ which queries are in the update set. Moreover, if $\mathcal{U}_{<t}$ is the set of all update queries from the first query sequence, and $\mathcal{U}'_{<t}$ is the set of all update queries from the second sequence, then $\mathcal{U}_{<t} \triangle \mathcal{U}'_{<t} = q^*$.

Now consider the distribution of H_t. Each sample in H_t comes from the distribution D_t, which is either

$$D_t(x) \propto \exp\left(-\frac{\eta}{s}\sum_{u \in \mathcal{U}_{<t}} u\right) \qquad \text{or} \qquad D'_t(x) \propto \exp\left(-\frac{\eta}{s}\sum_{u \in \mathcal{U}'_{<t}} u\right)$$

Given this, it is easy to see that for any x we have

$$\left|\ln\left(\frac{D_t(x)}{D'_t(x)}\right)\right| \leq 2\eta/s := \varepsilon_0.$$

Notice that once i_{t-1} and H_t are fixed, i_t depends only on the choice of s_t (the number of bad rounds to allow before updating the hypothesis), which is independent of the query sequence and thus incurs no additional privacy loss. Thus the only privacy loss comes from the $\widehat{n}$ samples in each of the T epochs, and the algorithm is a $\widehat{n}T$-fold adaptive composition of $(\varepsilon_0, 0)$ differentially private algorithms. A standard composition analysis (Theorem 2.3) shows that the components v are (ε', δ)-DP for $\varepsilon' = \varepsilon_0\sqrt{2\widehat{n}T\log(1/\delta)} + 2\varepsilon_0^2 T \leq \varepsilon/2$. This completes the proof of the claim. □

Combining these two claims proves the theorem. □

7.6.4 Handling Arbitrary Low-Sensitivity Queries

We can also modify this algorithm to answer arbitrary Δ-sensitive queries, albeit with worse accuracy bounds. As with our offline algorithms, we modify the algorithm to run the multiplicative weights updates over the set of databases $\mathcal{X}^n$ and adjust the parameters. When we run multiplicative weights over a support of size $|\mathcal{X}|^n$ (rather than $|\mathcal{X}|$), the number of epochs increases by a factor of n, which in turn affects the amount of noise we have to add to ensure privacy.

We will now sketch the argument, ignoring the parameters β and δ for simplicity. In order to get convergence of the multiplicative weights distribution, we need to take $T \approx \frac{n \log |\mathcal{X}|}{\eta^2}$ and in order to ensure that H_t approximates D_t sufficiently well, we take $\widehat{n} \approx \sqrt{\frac{\log k}{\eta^2}}$. Recall that to argue *analyst privacy*, we viewed the algorithm as being (essentially) the $\widehat{n}T$-fold composition of ε_0-analyst private algorithms, where $\varepsilon_0 = \eta/s$. In order to get analyst privacy, we needed

$$\frac{\eta}{s} \lesssim \frac{\varepsilon}{\sqrt{\widehat{n}T}} \approx \frac{\varepsilon\eta^2}{\sqrt{n \log |\mathcal{X}|} \log^{1/4} k}$$
$$\implies s \gtrsim \frac{\sqrt{n \log |\mathcal{X}|} \log^{1/4} k}{\varepsilon\eta}.$$

Once we have set s (as a function of the other parameters) to achieve analyst privacy, we can work on establishing data privacy. As before, the number of bad rounds will be

$$R \approx sT \approx \frac{n^{3/2} \log^{3/2} |\mathcal{X}| \log^{1/4} k}{\varepsilon\eta^3}.$$

Given this bound on the number of bad rounds, we need to set

$$\sigma \approx \frac{\Delta\sqrt{R}}{\varepsilon} \approx \frac{\Delta n^{3/4} \log^{3/4} |\mathcal{X}| \log^{1/8} k}{\varepsilon\eta^{3/2}}$$

to obtain data privacy, and

$$\tau \approx \sigma \log k \approx \frac{\Delta n^{3/4} \log^{3/4} |\mathcal{X}| \log^{9/8} k}{\varepsilon\eta^{3/2}}$$

to ensure that all the update queries truly have large error on the current hypothesis H_t.

The final error bound will come from observing that η and τ are both lower bounds on the error. The error is bounded below by τ because that is the noise threshold set by the algorithm, and τ must be larger than η or else we cannot argue that multiplicative weights makes progress during update rounds. Thus setting $\eta = \tau$ will approximately minimize the error.

The final error bound we obtain (ignoring the parameters β and δ) is

$$O\left(\frac{\Delta^{2/5} n^{3/10} \log^{3/10} |\mathcal{X}| \log^{9/20} k}{\varepsilon^{2/5}}\right),$$

which gives a non-trivial error guarantee when $\Delta \ll 1/n^{3/4}$.

7.7 A Secrecy-of-the-Sample Lemma

In this section we give a formal proof of Lemma reflem:sdtodp. First we restate the lemma. Consider the following process:

- Fix an (ε, δ)-differentially private algorithm $\mathcal{A}\colon \mathcal{U}^* \to \mathcal{R}$ and a bit $b \in \{0,1\}$. Let $D_0 = \emptyset$.
- For $t = 1, \ldots T$
  - The (possibly randomized) adversary $\mathcal{B}(y_1, \ldots, y_t; r)$ chooses two distributions B_t^0, B_t^1 such that $SD(B_t^0, B_t^1) \leq \sigma$.
  - Choose $x_t \leftarrow_{\text{R}} B_t^b$ and let $D_t = D_{t-1} \cup \{x_t\}$.
  - Choose $y_t \leftarrow_{\text{R}} \mathcal{A}(D_t)$.

For a fixed algorithm $\mathcal{A}$ and adversary $\mathcal{B}$, let V^0 be the distribution on $(y_1, \ldots, y_T)$ when $b = 0$ and V^1 be the distribution on $(y_1, \ldots, y_T)$ when $b = 1$.

Lemma 7.32. *If $\varepsilon \leq 1/2$ and $T\sigma \leq 1/12$, then with probability at least $1 - T\delta - \delta'$ over $y = (y_1, \ldots, y_T) \leftarrow_R V^0$,*

$$\left| \ln \left(\frac{V^0(y)}{V^1(y)} \right) \right| \leq \varepsilon(T\sigma)\sqrt{2T \log(1/\delta')} + 30\varepsilon^2(T\sigma)T.$$

Proof. Given distributions B^0, B^1 such that $SD(B^0, B^1) \leq \sigma$, there exist distributions C^0, C^1, C such that $B^0 = \sigma C^0 + (1-\sigma)C$ and $B^1 = \sigma C^1 + (1-\sigma)C$. An alternative way to sample from the distribution B^b is to flip a coin $c \in \{0,1\}$ with bias σ, and if the coin comes up 1, sample from C^b, otherwise sample from C.

Consider a partial transcript $(r, y_1, \ldots, y_{t-1})$. Fixing the randomness of the adversary will fix the coins $c_1, \ldots, c_T$, which determine whether or not the adversary samples from C_j^b or C_j for $j \in [T]$. Let $w = \sum_{j=1}^T c_j$. Fixing the randomness of the adversary and $y_1, \ldots, y_{t-1}$ will also fix the distributions C_j for $j \leq t$ and, in rounds for which $c_j = 0$, will fix the samples x_j for $j \leq t$. If we let D_t^0, D_t^1 denote the database D_t in the case where $b = 0, 1$, respectively, then we have

$$|D_t^0 - D_t^1| \leq \sum_{j=1}^t c_j \leq \sum_{j=1}^T c_j = w.$$

Thus,

$$\left|\ln\left(\frac{V_t^0(y_t|r,y_1,\ldots,y_{t-1})}{V_t^1(y_t|r,y_1,\ldots,y_{t-1})}\right)\right| \leq w\varepsilon,$$

and

$$\mathbb{E}\left[\ln\left(\frac{V_t^0(y_t|r,y_1,\ldots,y_{t-1})}{V_t^1(y_t|r,y_1,\ldots,y_{t-1})}\right)\right] \leq w\varepsilon\min\left\{e^{w\varepsilon}-1,1\right\},$$

where the expectation is taken over $V_t^0|r,y_1,\ldots,y_{t-1}$.

Fix $w \in \{0,\ldots,T\}$. Conditioning on any r such that $\sum_{t=1}^{T} c_t = w$, we can apply Azuma's inequality as in [35] to obtain

$$D_\infty^{T\delta+\delta'}(V^0|w||V^1|w) \leq w\varepsilon\sqrt{2T\log(1/\delta')} + w\varepsilon\min\left\{e^{w\varepsilon}-1,1\right\}T.$$

Thus,

$$\begin{aligned} D_\infty^{T\delta+\delta'}(V^0||V^1) &\leq \sum_{w=1}^{T}\Pr[w]\left(w\varepsilon\sqrt{2T\log(1/\delta')} + w\varepsilon\min\left\{e^{w\varepsilon}-1,1\right\}T\right) \\ &= \sum_{w=1}^{T}\Pr[w]\,w\varepsilon\sqrt{2T\log(1/\delta')} + \sum_{w=1}^{T}\Pr[w]\,w\varepsilon\min\left\{e^{w\varepsilon}-1,1\right\}T. \end{aligned} \tag{7.4}$$

First, we consider the left sum in (7.4).

$$\begin{aligned} &\sum_{w=1}^{T}\Pr[w]\,w\varepsilon\sqrt{2T\log(1/\delta')} \\ &= \varepsilon\sqrt{2T\log(1/\delta')}\sum_{w=1}^{T}\binom{T}{w}\sigma^w(1-\sigma)^{T-w}w \\ &= \varepsilon\sqrt{2T\log(1/\delta')}(T\sigma)\sum_{w=0}^{T-1}\binom{T-1}{w}\sigma^w(1-\sigma)^{T-1-w} && \left(\tbinom{T}{w}w = \tbinom{T-1}{w-1}T\right) \\ &= \varepsilon\sqrt{2T\log(1/\delta')}(T\sigma) \end{aligned}$$

Now, we work on the right sum in (7.4).

$$
\begin{aligned}
&\sum_{w=1}^{T} \Pr[w] \left(w\varepsilon \min\left\{e^{w\varepsilon}-1, 1\right\} T\right) \\
&= \sum_{w=1}^{T} \binom{T}{w} \sigma^w (1-\sigma)^{T-w} \left(w\varepsilon \min\left\{e^{w\varepsilon}-1, 1\right\} T\right) \\
&= \left(4\varepsilon^2 T\right) \sum_{w=1}^{1/\varepsilon} \binom{T}{w} \sigma^w (1-\sigma)^{T-w} w + (\varepsilon T) \sum_{w=1/\varepsilon}^{T} \binom{T}{w} \sigma^w (1-\sigma)^{T-w} w \\
&= \left(4\varepsilon^2 T\right) \sum_{w=1}^{1/\varepsilon} \left(\frac{eT\sigma}{w}\right)^w w^2 + (\varepsilon T) \sum_{w=1/\varepsilon}^{T} \left(\frac{eT\sigma}{w}\right)^w w \\
&\leq \left(4\varepsilon^2 T\right) \sum_{w=1}^{1/\varepsilon} (eT\sigma)^w + (\varepsilon T) \sum_{w=1/\varepsilon}^{T} (eT\sigma)^w && (w^2/w^w \leq 1 \text{ for } w \in \mathbb{N}) \\
&\leq 4\varepsilon^2 T(2eT\sigma) + 2(eT\sigma)^{-1/\varepsilon} \varepsilon T \leq 3\varepsilon^2 T && (eT\sigma \leq 1/4) \\
&\leq 24\varepsilon^2 (T\sigma)T + 4\varepsilon^2 (T\sigma)T \leq 30\varepsilon^2 (T\sigma)T
\end{aligned}
$$

Combining our bounds for the left and right sums in (7.4) completes the proof. □

Chapter 8

Conclusion

In this thesis, we have made several contributions to understanding the computational complexity of natural privacy-preserving data analysis tasks. We first examined the question of how many arbitrary counting queries can be answered by an efficient sanitizer. In Chapter 3, we answered this question by showing that, assuming the existence of one-way functions, there is no differentially private sanitizer that can accurate answer $n^{2+o(1)}$ arbitrary counting queries.

In light of the previous result, it was natural to ask whether or not it is possible to efficiently and privately answer many more than n^2 simple counting queries. In particular, we studied the computational complexity of answering marginal queries. In Chapter 4, we showed that computational complexity is still a major barrier even for marginal queries. Specifically, we showed that, assuming the existence of one-way functions, there is no efficient, differentially private one-shot sanitizer that generate private synthetic data preserving even the answers to all 2-way marginals. We also showed that if synthetic data is not required, then there exist sanitizers for marginal queries that are faster and require less data than previous approaches. In Chapter 5 we introduced a one-shot sanitizer for k-way marginals on a database with d attributes that has running time and minimum database size roughly $d^{O(\sqrt{k})}$, which is much less than the number of such queries. Then in Chapter 6 we considered one-shot sanitizers that have nearly optimal minimum database size. We showed that when $k \ll d$, such a one-shot sanitizer for k-way marginals exists with running time $2^{o(d)}$, improving on the 2^d running time required by the private multiplicative weights algorithm with only a slight decrease in utility.

Despite these results, there is still a significant gap in our knowledge between private-data-

analysis tasks for which we have efficient algorithms and tasks that we know are intractable. One important open problem is to design faster algorithms for answering k-way marginal queries. Arguably, the "holy grail" would be a one-shot sanitizer that is $.01$-accurate for all k-way marginal queries with running time $\mathrm{poly}(d, k, n)$ and minimum database size $\mathrm{poly}(d, k)$. We would like to call attention to one appealing step towards this goal that may be open to attack: find a differentially private one-shot sanitizer that is $.01$-accurate for all k-way marginal queries and has running time $\mathrm{poly}(d^k, n)$ and minimum database size $\mathrm{poly}(d, k)$.

Another significant open question is whether we can prove hardness results (that do not rely on synthetic data) for simple families of counting queries. Currently, the simplest family for which we know answering $n^{2+o(1)}$ counting queries is hard consists of depth-6 circuits of size $\mathrm{poly}(d, n)$, which is a family of size $\gg 2^n$. Dwork et al. [32] showed that, under certain cryptographic assumptions there is a family of size roughly $2^{\sqrt{n}}$ queries for which private and accurate one-shot sanitization is hard. An unsatisfying feature of these results is that the size of the query families in question depends on n. Most natural families of counting queries, such as marginals, have size that is independent of n, and depends only on d. Given our current state of knowledge, it is possible that for every family of queries of size $2^{\mathrm{poly}(d)}$, there exists an efficient one-shot sanitizer that guarantees accuracy on databases of size $\mathrm{poly}(d)$. However, we conjecture that such a statement is false. Thus, a major open problem is to find a (possibly not natural) family of counting queries of size $2^{\mathrm{poly}(d)}$ for which there is no private, efficient, and accurate one-shot sanitizer. Dwork et al. [32] showed that such a result would imply the existence of a certain weak form of traitor-tracing scheme with "constant-size ciphertexts" (length that depends only on the security parameter), which is a notorious open problem. However, recent advances in cryptography, in particular functional encryption, suggest that this open problem may be close to a resolution, which could in turn lead to new hardness-of-sanitization results.

In addition to studying the complexity of natural tasks in private data analysis, in Chapter 7 we designed new sanitizers that ensure differential privacy for both the data subjects and for the data analysts. In particular, we focused on sanitizers that guarantee privacy for the analyst even if all other analysts collude, and constructed sanitizers that answer exponentially many queries while ensuring the privacy of a single query asked by one analyst (one-query-to-many-analyst privacy) or all of the queries asked by one analyst (one-analyst-to-many-analyst privacy). In the former case, we were able to answer many arbitrary counting queries while achieving accuracy

$\widetilde{O}(1/\sqrt{n})$, which is the optimal dependence on n up to polylogarithmic factors. Although our analyst-private sanitizers achieved better accuracy than was previously known, they still do not match the accuracy achieved by differentially private sanitizers without a guarantee of analyst privacy. Thus, the main open question raised by the results of Chapter 7 is whether or not there is any gap between sanitizers satisfying some notion of analyst differential privacy and those that merely satisfy differential privacy for the data subjects.

Our belief is that the study of each of these open problems will lead to progress towards resolving the dilemma of privacy-preserving data analysis.

Bibliography

[1] *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*. IEEE Computer Society, 2009.

[2] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, 2008.

[3] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 805–816. ACM, 2012.

[4] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, Cambridge, UK, 2009.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[6] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[7] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *STOC*, pages 21–31. ACM, 1991.

[8] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In Leonid Libkin, editor, *PODS*, pages 273–282. ACM, 2007.

[9] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *SIAM J. Comput.*, volume 38, pages 1661–1694, 2008.

[10] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In Claire Mathieu, editor, *SODA*, pages 1193–1200. SIAM, 2009.

[11] M. Barbarao and T. Zeller. A face is exposed for AOL searcher 4417749. *The New York Times*, page Page A1, August 9, 2006.

[12] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. In *SIAM J. Comput.*, volume 36, pages 889–974, 2006.

[13] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[14] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill, New York, 1968.

[15] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In Chen Li, editor, *PODS*, pages 128–138. ACM, 2005.

[16] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In Cynthia Dwork, editor, *STOC*, pages 609–618. ACM, 2008.

[17] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer, 2006.

[18] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[19] Jehoshua Bruck and Roman Smolensky. Polynomial threshold functions, AC^0 functions, and spectral norms. *SIAM J. Comput.*, 21(1):33–42, 1992.

[20] Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov-bernstein inequalities. *CoRR*, abs/1302.6191, 2013.

[21] Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. *CoRR*, abs/1304.3754, 2013.

[22] Mahdi Cheraghchi, Adam Klivans, Pravesh Kothari, and Homin K. Lee. Submodular functions are noise stable. In Dana Randall, editor, *SODA*, pages 1586–1592. SIAM, 2012.

[23] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.

[24] Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. In *J. Comput. Syst. Sci.*, volume 51, pages 511–522, 1995.

[25] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[26] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003.

[27] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the pcp theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.

[28] Cynthia Dwork. The differential privacy frontier (extended abstract). In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 496–502. Springer, 2009.

[29] Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[30] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[31] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In Schulman [79], pages 715–724.

[32] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In Michael Mitzenmacher, editor, *STOC*, pages 381–390. ACM, 2009.

[33] Cynthia Dwork, Moni Naor, and Salil P. Vadhan. The privacy of the analyst and the power of the state. In *FOCS*, pages 400–409. IEEE Computer Society, 2012.

[34] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544. Springer, 2004.

[35] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE Computer Society, 2010.

[36] Vitaly Feldman. Hardness of proper learning. In *The Encyclopedia of Algorithms*. Springer-Verlag, 2008.

[37] Vitaly Feldman. A complete characterization of statistical query learning with applications to evolvability. In *FOCS* [1], pages 375–384.

[38] Vitaly Feldman and Pravesh Kothari. Learning coverage functions. *Manuscript*, 2013.

[39] Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332. ACM, 1996.

[40] Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[41] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 803–812. ACM, 2011.

[42] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 339–356. Springer, 2012.

[43] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 2348–2356, 2012.

[44] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70. IEEE Computer Society, 2010.

[45] Moritz Hardt, Guy N. Rothblum, and Rocco A. Servedio. Private data release via learning thresholds. In Dana Randall, editor, *SODA*, pages 168–187. SIAM, 2012.

[46] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In Schulman [79], pages 705–714.

[47] Johan Håstad. Some optimal inapproximability results. In *J. ACM*, volume 48, pages 798–859, 2001.

[48] Heritage Provider Network, Inc. Heritage Health Prize. `http://www.heritagehealthprize.com/`, 2011.

[49] Nils Homer. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 2008.

[50] Prateek Jain and Abhradeep Thakurta. Mirror descent based database privacy. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *APPROX-RANDOM*, volume 7408 of *Lecture Notes in Computer Science*, pages 579–590. Springer, 2012.

[51] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[52] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *J. ACM*, volume 41, pages 67–95, 1994.

[53] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[54] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In Tomas Sander, editor, *Digital Rights Management Workshop*, volume 2320 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2001.

[55] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In Kosaraju et al. [59], pages 723–732.

[56] Gary King. The changing evidence base of social science research. *The Future of Political Science: 100 Perspectives*, 2009.

[57] Hartmut Klauck. On arthur merlin games in communication complexity. In *Proceedings of the 26th Annual Conference on Computational Complexity (CCC)*, CCC, pages 189–199, 2011.

[58] Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. In John Shawe-Taylor and Yoram Singer, editors, *COLT*, volume 3120 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2004.

[59] S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors. *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*. ACM, 1992.

[60] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In Jan Paredaens and Dirk Van Gucht, editors, *PODS*, pages 123–134. ACM, 2010.

[61] Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, 2012.

[62] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2008.

[63] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute, 2011.

[64] Silvio Micali. Computationally sound proofs. In *SIAM J. Comput.*, volume 30, pages 1253–1298, 2000.

[65] Ilya Mironov. On significance of the least significant bits for differential privacy. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM Conference on Computer and Communications Security*, pages 650–661. ACM, 2012.

[66] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *STOC*, pages 33–43. ACM, 1989.

[67] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy*, pages 300–314. IEEE Computer Society, 2012.

[68] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. *CoRR*, abs/1102.4374, 2011.

[69] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[70] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. *CoRR*, abs/1212.0297, 2012.

[71] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *J. Comput. Syst. Sci.*, volume 43, pages 425–440, 1991.

[72] Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In Kosaraju et al. [59], pages 468–474.

[73] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. In *J. ACM*, volume 35, pages 965–984, 1988.

[74] A. Rakhlin and K. Sridharan. Statistical Learning Theory and Sequential Prediction. `http://www-stat.wharton.upenn.edu/~rakhlin/courses/stat928/`, 2012.

[75] John Rompel. One-way functions are necessary and sufficient for secure signatures. In Harriet Ortiz, editor, *STOC*, pages 387–394. ACM, 1990.

[76] Aaron Roth. Differential privacy and the fat-shattering dimension of linear queries. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 6302 of *Lecture Notes in Computer Science*, pages 683–695. Springer, 2010.

[77] Aaron Roth. The algorithmic foundations of data privacy, course notes. `http://www.cis.upenn.edu/~aaroth/courses/privacyF11.html`, 2011.

[78] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In Schulman [79], pages 765–774.

[79] Leonard J. Schulman, editor. *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 2010.

[80] Rocco Servedio, Li-Yang Tan, and Justin Thaler. Attribute-efficient learning and weight-degree tradeoffs for polynomial threshold functions. In *25th Annual Conference on Computational Learning Theory (COLT), JMLR Workshop and Conference Proceeding*, volume 23, pages 14.1–14.19, 2012.

[81] Alexander A. Sherstov. Approximate inclusion-exclusion for arbitrary symmetric functions. *Computational Complexity*, 18(2):219–247, 2009.

[82] Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. In *FOCS* [1], pages 343–362.

[83] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[84] Daniel Spielman. *Computationally Efficient Error-Correcting Codes and Holographic Proofs*. PhD thesis, MIT, Cambridge, MA, 1995.

[85] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[86] Gábor Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), 2008.

[87] Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. Faster algorithms for privately releasing marginals. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 810–821. Springer, 2012.

[88] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. *STOC 2013 (To appear)*.

[89] Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2011.

[90] Salil Vadhan. Personal communication. March 2013.

[91] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[92] Robert Špalek. A dual polynomial for OR. *CoRR*, abs/0803.4516, 2008.